Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations	1. Thesis and Dissertation Collection, all items

1991-09

Use of Hopfield networks for system identification and failure detection in autonomous underwater vehicles

Marsilio, Alan M.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/28628

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USE OF HOPFIELD NETWORKS FOR SYSTEM IDENTIFICATION AND FAILURE DETECTION IN AUTONOMOUS UNDERWATER VEHICLES

by

Alan M. Marsilio

SEPTEMBER 1991

Thesis Advisor: A.J. Healey

Approved for public release: Distribution is unlimited

Unclassified

SECUPITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release: Distribution is unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Naval Postgraduate School

6b. OFFICE SYMBOL (If applicable)
ME

7a. NAME OF MONITORING ORGANIZATION
Naval Postgraduate School

6c. ADDRESS (City, State and ZIP Code)
Monterey, CA 93943-5000

7b. ADDRESS (City, State, and ZIP Code)
Monterey, CA 93943-5000

8a. NAME OF FUNDING SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBER

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
USE OF HOPFIELD NETWORKS FOR SYSTEM IDENTIFICATION AND FAILURE DETECTION IN AUTONOMOUS UNDERWATER VEHICLES

12. PERSONAL AUTHORS
ALAN M. MARSILIO

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM_____ TO_____	SEPTEMBER 1991	99

16. SUPPLEMENTARY NOTATION
The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block numbers)
FIELD	GROUP	SUB-GROUP	Neural Network, Hopfield Network, Failure Dectection, System Parameter Identification, Autonomous Underwater Vehicles

19. ABSTRACT (Continue on reverse if necessary and identify by block numbers)

In the early 1980's John J. Hopfield developed a recurrent network based on a model of biological neurons. In his model, each neuron accepts inputs from all other neurons in the network, modifies each input with a weight and converts their sum to an output via the non-linear sigmoid transfer function. This output is then fed back to each of the input paths where the input signals are updated before the next summation. It has been proposed that this network can be successfully applied to the problem of system parameter identification where the weights are functions of the system states and the network, after being allowed to process a continuous block of system states, is guaranteed to converge to the system parameters. This thesis explores the concepts of network stability and solution existence for a time-invariant system. It is shown that the network will converge as expected provided the steady-state solution falls within the range of values of the sigmoid transfer function. Experimentation with the network when not all system states are measurable revealed that knowledge of the actual system parameters is necessary to obtain convergence because of large error between the actual and estimated system states, showing that minimization of this error must take place before the network is integrated. Finally, it is shown that as system parameters vary, the Hopfield network will track the parameter changes provided the system remains persistently excited by the input.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
XX UNCLASSIFIED/UNLIMITED __ SAME AS RPT __ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
A.J. Healey	(408) 646-2586	ME/Hy

DD Form 1473, JUN 86 *Previous editions are obsolete.* S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Approved for public release: Distribution is unlimited

Use of Hopfield Networks for System Identification and Failure Detection in Autonomous Underwater Vehicles

by

Alan M. Marsilio
Lieutenant, United States Coast Guard
B.S., Electrical Engineering, U.S. Coast Guard Academy, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

SEPTEMBER 1991

ABSTRACT

In the early 1980's John J. Hopfield developed a recurrent network based on a model of biological neurons. In his model, each neuron accepts inputs from all other neurons in the network, modifies each input with a weight and converts their sum to an output via the non-linear sigmoid transfer function. This output is then fed back to each of the input paths where the input signals are updated before the next summation. It has been proposed that this network can be successfully applied to the problem of system parameter identification where the weights are functions of the system states and the network, after being allowed to process a continuous block of system states, is guaranteed to converge to the system parameters. This thesis explores the concepts of network stability and solution existence for a time-invariant system. It is shown that the network will converge as expected provided the steady-state solution falls within the range of values of the sigmoid transfer function. Experimentation with the network when not all system states are measurable revealed that knowledge of the actual system parameters is necessary to obtain convergence because of large error between the actual and estimated system states, showing that minimization of this error must take place before the network is integrated. Finally, it is shown that as system parameters vary, the Hopfield network will track the parameter changes provided the system remains persistently excited by the input.

TABLE OF CONTENTS

I. INTRODUCTION 1
A. BACKGROUND OF THE NPS AUV II PROJECT 1
B. ARTIFICIAL NEURAL NETWORKS 2
C. ANN AND NPS AUV II 3
II. PARAMETER IDENTIFICATION AND HOPFIELD NETWORKS 6
A. INTRODUCTION 6
B. HISTORY OF ANN 6
C. HOPFIELD'S CONTRIBUTION 11
D. THE ISSUE OF STABILITY 13
E. THE CONTINUOUS HOPFIELD MODEL 15
F. THE HOPFIELD NETWORK FOR PARAMETER IDENTIFICATION 17
III. RESULTS OF THE HOPFIELD SIMULATION 24
A. INTRODUCTION 24
B. LINEAR TRANSFER FUNCTION 24
C. NON-LINEAR, SIGMOID, TRANSFER FUNCTION . . . 34
D. REMARKS 42
IV. PARAMETER IDENTIFICATION USING STATE OBSERVERS . . 46
A. INTRODUCTION 46
B. FULL-STATE OBSERVER 46
C. REDUCED-ORDER OBSERVER 54
D. REMARKS 59
V. TRACKING OF VARYING SYSTEM PARAMETERS 62
A. INTRODUCTION 62
B. RESULTS 62
VI. SUMMARY AND CONCLUSION 68
A. SUMMARY 68
B. CONCLUSION 70
C. RECOMMENDATIONS 71

APPENDIX A: PROOF OF STABILITY OF THE HOPFIELD NETWORK FOR PARAMETER IDENTIFICATION . . 73
APPENDIX B: MAIN PROGRAM FOR HOPFIELD NETWORK WITH LINEAR ACTIVATION FUNCTION 76
APPENDIX C: MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION 78
APPENDIX D: MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION AND FULL-STATE OBSERVER 80
APPENDIX E: MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION AND REDUCED-ORDER OBSERVER 82
APPENDIX F: MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION TRACKING VARIATION IN GAIN PARAMETER B(1) 84
LIST OF REFERENCES 87
INITIAL DISTRIBUTION LIST 89

LIST OF TABLES

TABLE I. THE "EXCLUSIVE-OR" PROBLEM 10

TABLE II. COMPARISON BETWEEN ACTUAL SYSTEM PARAMETERS AND SOLUTION USING FULL-STATE OBSERVER 54

TABLE III. COMPARISON BETWEEN ACTUAL PARAMETERS AND SOLUTION FOR REDUCED-ORDER OBSERVER 56

TABLE IV. EIGENVALUES OF W WHILE TRACKING [B(1)] . . 66

TABLE V. STEADY STATE VALUES OF V ($V_{ss} = W^{-1}*I'$) . . 66

LIST OF FIGURES

Figure 1. Single Perceptron Network 7
Figure 2. Typical Recurrent Network 12
Figure 3. Sigmoid Function for $\lambda = 0.1$ 16
Figure 4. Block Diagram of Continuous Hopfield Network 22
Figure 5. Test Case System, State Variable Response 26
Figure 6. Singular Values of W 28
Figure 7. Normalized Steady-State Vector, "thss" . . 30
Figure 8. Normalized A and B Matrix Coefficients, Linear Function 32
Figure 9. Non-Dimensional Norm of Theta 33
Figure 10. Sigmoid Function as λ Increases From 0.01 to 0.1 36
Figure 11. Sigmoid Function as Gain, G, Increases From 10 to 50, $\lambda = 0.1$ 37
Figure 12. Output of Network Using Sigmoid Function . 38
Figure 13. Sigmoid Function for λ From 0.01 to 0.1, with $G = 2/\lambda$ 41
Figure 14. Normalized A and B Matrix Coefficients, Sigmoid Function 43
Figure 15. Normalized Output Vector 44
Figure 16. Convergence of System Parameters for Full-State Observer 49
Figure 17. Normalized Output Vector for Full-State Observer 50
Figure 18. Normalized Solution When Parameters Imperfectly Estimated, Full State 52
Figure 19. Actual and Estimated System Response, Full State 53
Figure 20. Normalized Solution Vector for Reduced-Order Observer 57

Figure 21. Normalized Solution Vector for Imperfectly Estimated Parameters, Reduced Order 58
Figure 22. Actual and Estimated Responses, Reduced Order 60
Figure 23. Tracking of Varying Gain Parameter [B(1)] 64

I. INTRODUCTION

A. BACKGROUND OF THE NPS AUV II PROJECT

The Naval Postgraduate School's Autonomous Underwater Vehicle II (NPS AUV II) supports the second generation of projects focusing on the development of an unmanned, untethered vehicle possessing sufficient, self-contained intelligence to perform a broad range of missions while being able to respond to unplanned situations and take appropriate actions. The project is part of the U.S. Navy's ongoing studies of unmanned, sub-surface, marine vehicles and their usefulness in an expanded role in the future Navy. As described in Healey *et al.*, [Ref. 1], the NPS AUV II is the first of its kind to attempt a Mission Planning Expert System which will serve as a framework within which all of the vehicle's logical operations and resulting actions will be carried out.

The vehicle's missions might consist of any of the sub-surface tasks currently being conducted by manned surface and sub-surface vehicles and by free-swimming personnel. It would be required to successfully navigate within a prescribed operating area while avoiding all stationary and moving obstacles but at the same time completely surveying any objects which meet the "special interest" criteria of its

mission planning system. Its mission would also include storing and analyzing data before deciding future courses of action appropriate to its mission and be capable of eventually down-loading all such data for human analysis. Unplanned situations might encompass sudden changes in the vehicle's operating environment such as shifts in current direction and speed, the sudden presence of belligerent animals or vehicles, faults in its own logic or operating systems, or errors in its mission program (i.e., incorrect navigation information).

The capabilities of such a vehicle are far removed from those of any similar platform in use today. Such a vehicle as NPS AUV II must ultimately possess complete "knowledge" of its own operational abilities, similar to what humans refer to as "motor skills," while at the same time be able to "think" about various responses and courses of action and decide which would best suit the goals of its mission. Such capability, and the numbingly vast array of possibilities it entails, can not possibly be programmed into a computer as a series of tasks. The prospect of a so-called "thinking machine" is not so far-fetched, however, and this is where the theory of Artificial Neural Networks (ANN) may eventually find application.

B. ARTIFICIAL NEURAL NETWORKS

The origin of ANN may be traced back to as early as 1943, when McCulloch and Pitts wrote their landmark paper "A Logical Calculus of Ideas Immanent in Nervous Activity" [Ref. 2].

Since then scientists, engineers, physicists, and biologists have been studying ways to mathematically model the human brain's ability to accept inputs from many, completely different types of sensors, analyze that data, decide on a course of action, trigger the proper response in its operational appendages, and learn from the results of that response. The focus of the research efforts in the field has been centered on a mathematical relationship called a neuron, akin to the biological brain cell of the same name, whose output is a weighted summation of the inputs from other neurons and which is then used either as an input to other neurons or as part of the output of the network as a whole. Most importantly, for application to "thinking machines," the output of any particular neuron can be used in a feedback loop to modify the weights associated with its own inputs. This type of network, known as "recurrent," shows a minimal ability to "learn" that a particular pattern of inputs produces a corresponding series of outputs based on the values of the input weights, marginally like the functioning of short-term memory in humans.

C. ANN AND NPS AUV II

In examining the need for NPS AUV II to "know" its own capabilities and limitations and the usefulness of ANN in "learning" a pattern associated with a particular input environment, a connection can be made with neural networks for

diagnostics [Ref. 3] and system parameter identification [Ref. 4]. It has been proposed that a Hopfield network [Ref.5] [Ref. 6] can be configured in continuous time to accept a time history of the state-space response of a dynamic system represented by the equation

$$\dot{x} = \mathbf{A}x + \mathbf{B}u \qquad (1)$$

where $\underline{x}$ is a vector of state variables, $\dot{\underline{x}}$ is its time derivative, and $\underline{u}$ is the system input, and identify the system parameters represented by the matrices **A** and **B**. System parameter identification provides the vehicle with its own blueprint for input-output response upon which it will base its decisions regarding the proper actions needed to effect desired results. By continuously updating its own database as system parameters change, such events as internal faults or external, environmental limitations can be detected and diagnosed.

The focus of this thesis is on exploring the method of Shoureshi and Chu [Ref. 4] to determine if it might, indeed, be useful for vehicle system parameter identification in real time. The next chapter provides a brief history of ANN, focusing on the Hopfield recurrent network and how it is adapted by Shoureshi and Chu to system parameter identification. Chapter III presents the results of stability investigations of the Hopfield network formulation and the speed of convergence to expected solution values. Chapter IV

delves into study of the use of full-state and reduced-order observers with the Hopfield network to determine if the network will correctly identify system parameters when not all system states are measurable. Chapter V is a study of the ability of the network to track the pattern of system parameters as they vary with time. Finally, in Chapter VI a summary of the strengths and limitations of Hopfield networks and recommendations for future research in related fields are presented.

II. PARAMETER IDENTIFICATION AND HOPFIELD NETWORKS

A. INTRODUCTION

This chapter begins with a brief history of the study of ANN and presents an overview of the first ANN algorithms, which continue to provide a foundation for current research. Emphasis is placed on the development and theory of Hopfield networks, not only because the Hopfield model is the basis for the research in this thesis, but because it also helped infuse the flagging ANN research community with new energy in the early 1980's. Much more detailed analyses of the entire history and scope of ANN research, including that which has been conducted since Hopfield's efforts in 1982-1984, can be found throughout Wasserman [Ref. 7] and NeuralWare [Ref. 8]. Those references also serve to guide the history and background information in this chapter.

B. HISTORY OF ANN

As mentioned in Chapter I the impetus for the study of mathematical models for brain activity began with McCulloch and Pitts in 1943. Their subsequent work focused on the neuron model shown in Figure 1, reprinted from Wasserman [Ref. 7:p. 28], which shows the neuron modeled as a summation of several weighted inputs. The result of the summation is then compared to some threshold value: if the threshold is exceeded, the

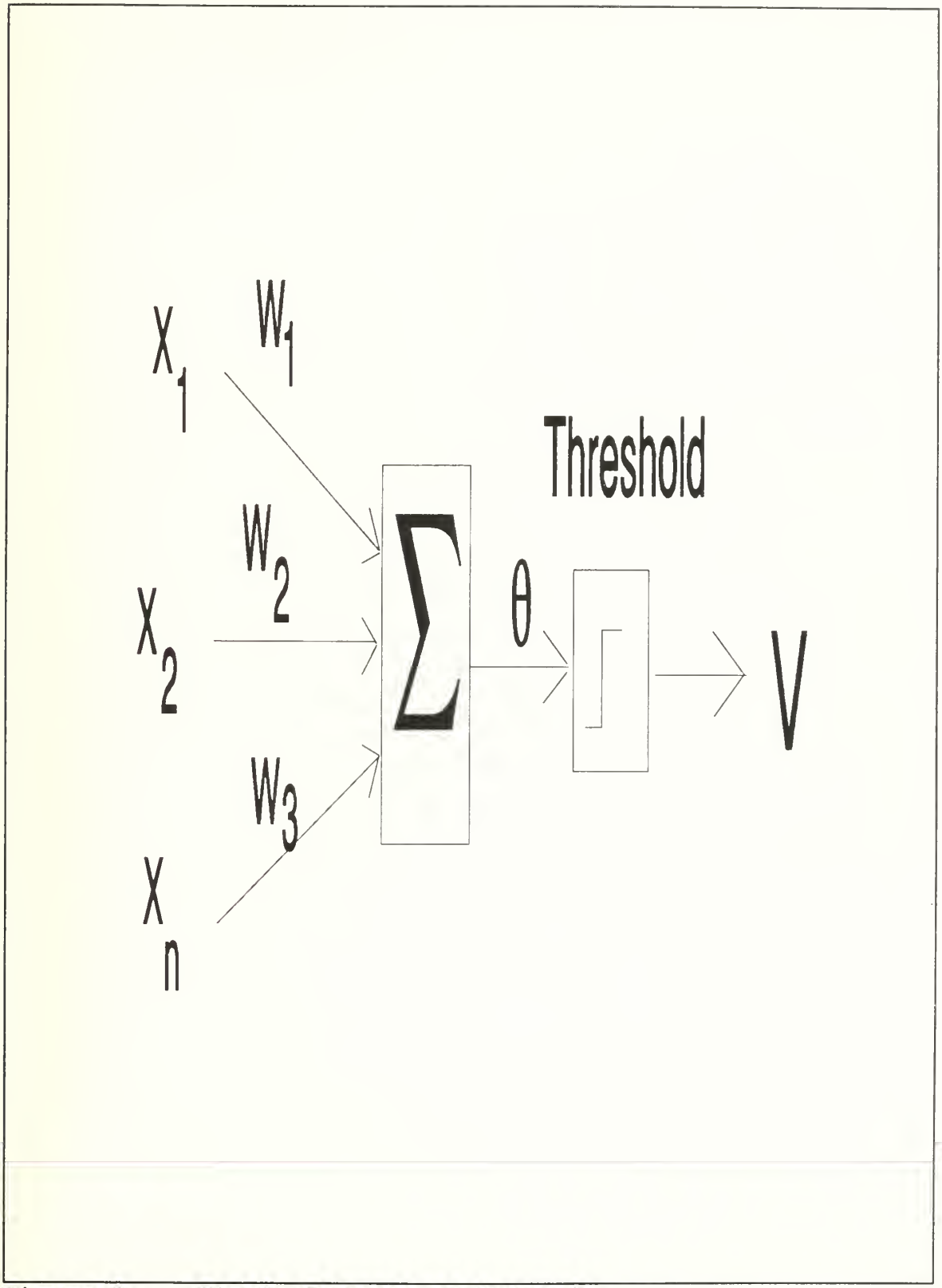

Figure 1. Single Perceptron Network

output is 1; otherwise it is 0. This basic system, which became the foundation on which the first generation of ANN was built, is represented mathematically as

$$V = \begin{cases} 1, & \text{if } \sum_{i=1}^{N} W_i x_i > \text{THRESHOLD} \\ 0, & \text{if } \sum_{i=1}^{N} W_i x_i < \text{THRESHOLD} \end{cases} \quad (2)$$

In the 1950's and 1960's this simple system was greatly expanded into multi-layered networks, with variations being applied to such diverse fields as pattern classification (Rosenblatt [Ref. 9]), signal filtering (Widrow [Ref. 10]), and macroscopic intelligence (Minsky [Ref. 11]). All such applications were based on a learning rule proposed by Hebb [Ref. 12] which assumes a multi-layered network where neurons are interconnected as well as available to receive and transmit external inputs and outputs. Basically, where two neurons, represented by subscripts i and j, are connected to each other and to other neurons in the network, the value of the weighted connection is adjusted based on the values, or "excitation levels," of the two neurons. In equation form,

$$W_{ij}(t+1) = W_{ij}(t) + \Theta_i \Theta_j \quad (3)$$

where W_{ij} ≡ the weight of connection from i to j
Θ_i ≡ the excitation level of neuron i
Θ_j ≡ the excitation level of neuron j

The weights are increased, and the connections strengthened, each time the vector of inputs produces excitation levels for

both i and j equal to the binary value 1. As each successive vector of training inputs is applied, a pattern of weight values emerges where often-used paths are strengthened well above the levels of little-used paths. Provided the training data is of sufficient quantity, this pattern will theoretically produce a desired vector of output values equivalent to those of the training sets and the network can be said to have learned the correct response to a given input.

In subsequent uses of Hebbian learning, the manner in which the products of the two excitation levels is obtained has been altered through various functions. Wasserman [Ref. 7:pp. 99-100] uses the term "activation function" to describe a class of functions where $\mathbf{V}=F(\Theta)$. The function F may be a simple threshold function where

$$\mathbf{V} = \begin{cases} 1, & \text{if } \Theta > \text{THRESHOLD} \\ 0, & \text{if } \Theta < \text{THRESHOLD} \end{cases} \qquad (4)$$

and THRESHOLD is some constant value. Later variations have modeled F as a simple linear function with or without some gain multiplier, while more recent forms have sought to emulate the actual activity of a biological neuron. This last form figures prominently in the work of Hopfield [Ref. 5] [Ref. 6] and will be the subject of extensive discussion in Section E of this chapter.

The claims that the proponents of ANN were making regarding the usefulness and wide application of their

algorithms led Minsky to apply rigid mathematical tests to the theory of "perceptrons," a term coined by Rosenblatt [Ref. 9] to describe his variation of the McCulloch-Pitts algorithm. In their book *Perceptrons* [Ref. 11] Minsky and Papert proved that the perceptron algorithm was not able to solve the simple, "exclusive-or" (XOR) problem, which is linearly inseparable. A detailed analysis of this shortcoming is presented by Wasserman [Ref. 7:pp. 29-33], who represents the XOR problem in tabular form for a perceptron consisting of one neuron and two inputs, designated x and y. Table I is reproduced here, where the values of x and y are binary and the output, Θ, follows the XOR rule.

TABLE I. THE "EXCLUSIVE-OR" PROBLEM

x	y	**V**
0	0	0
1	0	1
0	1	1
1	1	0

The equation for the summation of the neuron inputs is

$$\Theta = xw_1 + yw_2 \tag{5}$$

where the output **V** of the neuron takes the form of Equation (4) where THRESHOLD can be any constant value between 0 and 1. Minsky proved that there is no set of weights, w_1 and w_2, that will completely duplicate Table I, regardless of the threshold

value. This revelation, among others, proved a heavy blow to the study of ANN and greatly contributed to the dearth of research in the field throughout the 1970's.

C. HOPFIELD'S CONTRIBUTION

Though limited work on ANN continued following Minsky's book, the field was fully revived in 1982 after a presentation by John J. Hopfield to the National Academy of Sciences in 1982 [Ref. 5]. Hopfield began with the system formulation proposed by McCulloch and Pitts, where the basic network consists of a set of neurons which compute the weighted sum of the inputs, then set the output to zero or one depending on their relation to a set threshold value. What made the Hopfield network unique, however, is the fact that the output of each neuron is fed back to the inputs of all other neurons. (In the original formulation, Hopfield believed a neuron could not be fed back to itself. This has since been abandoned as a condition.) An example of this "recurrent" network is shown in Figure 2, reprinted from Wasserman [Ref. 7:p. 95]. The activity of a neuron is represented as

$$\Theta_j = \sum_{i=1}^{N} W_{ij} V_i + I_j \tag{6}$$

where the neuron output

$$V_j = \begin{cases} 1, & \text{if } \Theta_j > \text{THRESHOLD}_j \\ 0, & \text{if } \Theta_j < \text{THRESHOLD}_j \end{cases} \tag{7}$$

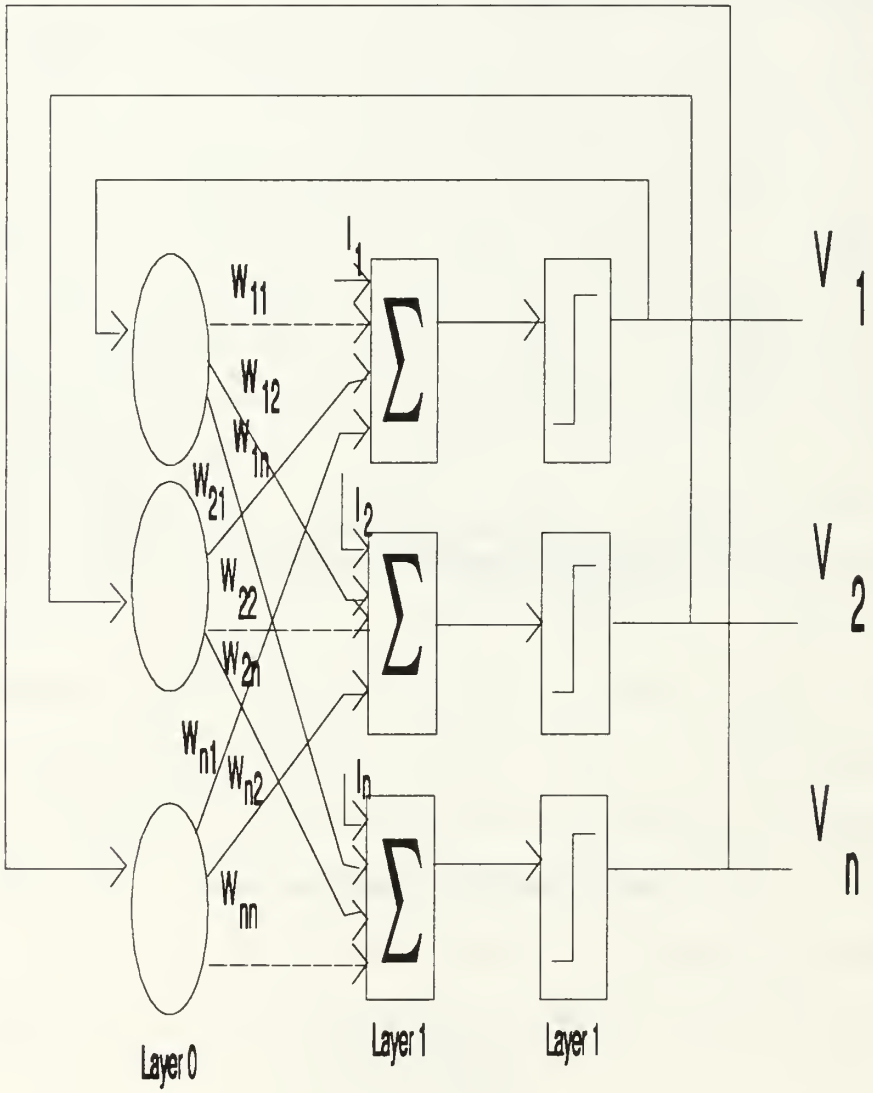

Figure 2. Typical Recurrent Network

Equation (6) is an expanded version of the original algorithm, where $\mathbf{I}_j$ is an external bias input. This form of the equation, and others that follow, appear in a subsequent paper presented by Hopfield to the National Academy of Sciences [Ref. 6]. The basic Hopfield learning rule, contrasted with that of Hebb, is

$$\Delta \mathbf{W}_{ij} = \sum_{i=1}^{N} \sum_{j=1}^{N} (2\mathbf{V}_i - 1)(2\mathbf{V}_j - 1) \qquad (8)$$
$$\Delta \mathbf{W}_{ij} = \Delta \mathbf{W}_{ji}$$

In Equation (8), $\mathbf{V}_i$ are values at a k^{th} iteration level in the training (learning) process and $\Delta \mathbf{W}_{ij}$ updates $\mathbf{W}_{ij}$ from k to k+1 as $k \rightarrow \infty$. It can be seen that the connection weights increase when the output of a neuron is the same as the input but the weight values decrease when the input and output differ.

D. THE ISSUE OF STABILITY

A major question which arose regarding the recurrent Hopfield network was that of stability. This clearly had not been a problem with perceptrons because of their feed-forward, static architecture. With the Hopfield net, however, a proof was needed that the network would converge to a stable state for all inputs. Building on work by Cohen and Grossberg [Ref. 13], Hopfield theorized that his model followed the activity of the bounded energy function represented by

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{W}_{ij} \mathbf{V}_i \mathbf{V}_j - \sum_{i=1}^{N} \mathbf{I}_i \mathbf{V}_i + \sum_{i=1}^{N} g_i \mathbf{V}_i \qquad (9)$$

where g is a set threshold value, $\mathbf{I}_i$ is a bias, and the energy of the network must decrease or remain the same as it changes state. Thus, the change in energy due to a change in the value of the neuron state $\mathbf{V}_i$ is

$$\Delta E = -\Delta \mathbf{V}_i \left[\sum_{j=1}^{N} \mathbf{W}_{ij} \mathbf{V}_j + \mathbf{I}_i - g_i \right] \qquad (10)$$

It can be seen that the sign of the expression in the brackets is inversely proportional to the change in energy E. Substituting Equation (6) into Equation (10) yields

$$\Delta E = -\Delta \mathbf{V}_i \left[\sum_{j=1}^{N} \Theta_i - g_i \right] \qquad (11)$$

or, if Θ_i is greater than the threshold g_i, the bracketed value is positive and the output of neuron i must change in the positive direction or remain constant. This means that $\Delta \mathbf{V}_i$ can only be positive or zero, so ΔE must be negative, or E is decreasing. If $\Theta_i < g_i$, the bracketed value is negative and $\Delta \mathbf{V}_i$ must be negative or zero so again ΔE must be negative. Finally, if $\Theta_i = g_i$, ΔE is zero and E remains constant. Since under all conditions E is either decreasing or unchanged, this function, which is bounded, must eventually reach some bounded minimum value, proving the eventual stability of the network states.

E. THE CONTINUOUS HOPFIELD MODEL

Having proven that his discrete-state model was stable under all conditions, Hopfield sought to expand the model to cover continuous systems, which represented a more realistic application [Ref. 6]. The original model, where the change in states of a neuron is represented by a binary "on-off" firing, was retained for the continuous formulation. However, Hopfield noted that actual state changes were non-linear and lagged the outputs of other neurons from which its inputs were fed. He modeled this characteristic as a simple RC circuit, with input capacitance C, neuron resistance R, and the impedance between the output of neuron j and input of neuron i designated $\mathbf{W}_{ij}$. Thus,

$$C_i \frac{d\Theta_i}{dt} = \sum_{j=1}^{N} \mathbf{W}_{ij}\mathbf{V}_j - \frac{\Theta_i}{R_i} + \mathbf{I}_i \qquad (12)$$

where

$$\Theta_i = g_i^{-1}(\mathbf{V}_i) \qquad (13)$$

The input-output relation represented by the function g_i is the non-linear "sigmoid" function

$$g(\Theta) = \frac{1}{1+e^{-\lambda\Theta}} \qquad (14)$$

where λ is the learning rate, a coefficient which determines the slope of the linear portion of the sigmoid function. Figure 3 is a plot of the function for λ=0.1.

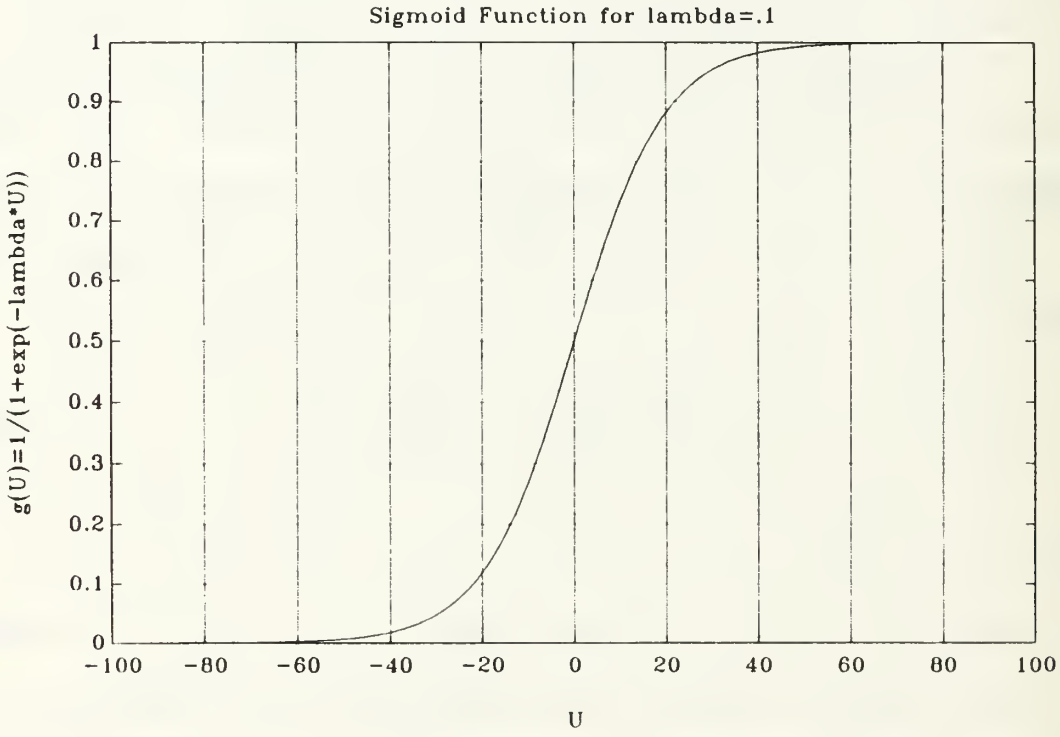

Figure 3. Sigmoid Function for $\lambda = 0.1$

For the purposes of determining stability the energy function from Equation (9) now becomes

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \mathbf{W}_{ij}\mathbf{V}_i\mathbf{V}_j + \sum_{i=1}^{N}\left(\frac{1}{R_i}\right)\int_0^{\mathbf{V}_i} g_i^{-1}(\mathbf{V})\,d\mathbf{V} + \sum_{i=1}^{N}\mathbf{I}_i\mathbf{V}_i \qquad (15)$$

The derivative with respect to time is

$$\frac{dE}{dt} = -\sum_{i=1}^{N}\frac{d\mathbf{V}_i}{dt}\left[\sum_{j=1}^{N}\mathbf{W}_{ij}\mathbf{V}_j - \frac{\Theta_i}{R_i} + \mathbf{I}_i\right] \qquad (16)$$

Substituting Equations (12) and (13) into Equation (16),

$$\frac{dE}{dt} = -\sum_{i=1}^{N} C_i g_i^{-1'}(\mathbf{V}_i)\left(\frac{d\mathbf{V}_i}{dt}\right)^2 \qquad (17)$$

Since $g_i^{-1}(\mathbf{V}_i)$ is a monotonically increasing function and C_i is positive, then all terms in the right-hand side of Equation (17) are positive and dE/dt is negative. If dE/dt = 0, this implies that $d\mathbf{V}_i/dt = 0$ and the neuron is not changing state, so a minimum energy has been reached. Hopfield's stability proof, however, is not too convincing as the energy function in this case is not necessarily bounded and can, under circumstances to be shown, diverge.

F. THE HOPFIELD NETWORK FOR PARAMETER IDENTIFICATION

Hopfield's method of modelling his network as the minimization of an energy function was adapted by Shoureshi and Chu [Ref. 4] for use in minimizing the equation error for system parameter identification. This has far-reaching

application to the control system field because having full knowledge of a system's dynamics in the form of parameter identification is essential for stable control, and leads to adaptive systems if parameters change.

The state-space form of any continuous system is represented by

$$\begin{aligned} \dot{\underline{x}}(t) &= \mathbf{A}\underline{x}(t) + \mathbf{B}\underline{u}(t) \\ \underline{y}(t) &= \mathbf{C}\underline{x}(t) \end{aligned} \tag{18}$$

where $\underline{x}(t)$ is a vector of system variables, or states; $\dot{\underline{x}}(t)$ is their time derivative vector, $\underline{u}(t)$ is some vector of time-varying inputs to the system, and $\underline{y}(t)$ is a vector of measurable outputs from the system response. The coefficients **A**, **B**, and **C**, are matrices which define the physical characteristics of the system. These matrices may depend on time, but will be modeled here as time-invariant. The equation error associated with incorrect estimates of the parameters when $\underline{u}(t)$, $\underline{x}(t)$, and $\dot{\underline{x}}(t)$ are fully measured, becomes

$$\dot{\underline{e}}(t) = \dot{\underline{x}}(t) - \mathbf{A}\underline{x}(t) - \mathbf{B}\underline{u}(t) \tag{19}$$

The goal of successful system identification is to minimize the error between the estimated and actual system parameters. A convenient method is to minimize a positive-definite function defined as the square of the equation error of the system. Such a function might be

$$J = \mathrm{tr}[E\{\underline{\dot{e}}(t)\cdot\underline{\dot{e}}'(t)\}] \tag{20}$$

Because of the time-variance of $\underline{\dot{e}}(t)$, expected values are used to formulate J, and the trace of the matrix of error values is taken as an average of the total error. Thus, after substitution of Equation (19) into (20)

$$J = E\{\mathbf{V}'(H'H)\mathbf{V} - 2\mathbf{V}'H'\underline{\dot{x}} + \underline{\dot{x}}'\underline{\dot{x}}\} \tag{21}$$

where

$$\mathbf{H} = \mathrm{diag}\{\underline{x}'_i(t), \underline{u}'_i(t)\}, \quad i = 1,n; \quad \mathbf{H} \in R^{n \times (n^2+nr)} \tag{22}$$

and

$$\mathbf{V}' = [a_1, b_1, \ldots a_i, b_i, \ldots a_n, b_n] \tag{23}$$

and a_i, b_i are the i^{th} rows of **A** and **B**.

A conceptual and mathematical similarity between J and the energy function E of Hopfield's papers can now be seen. The point is to minimize the error function J to produce a least-squares equation error of the continuous control system, much as Hopfield showed that his network minimized the energy function E. Furthermore, identification of the proper system parameters requires that minimization of J be conducted with respect to the vector of neuron outputs **V**. Calculating that partial derivative yields

$$\frac{\partial J}{\partial \mathbf{V}} = E\{(\mathbf{H}'\mathbf{H})\mathbf{V}\} - E\{\mathbf{H}'\underline{\dot{x}}\} \tag{24}$$

At this point, a direct connection is made between the state-space form of the system dynamics and the Hopfield model. Equating Equation (25) with Equation (12) and assuming $R_i \rightarrow \infty$ and $C_i=1$,

$$\frac{\partial J}{\partial \mathbf{V}} = -\frac{d\Theta_i}{dt} = \sum_{j=1}^{N} \mathbf{W}_{ij}\mathbf{V}_j - \mathbf{I}_i \qquad (25)$$

Now, in comparing Equations (25) and (24) it can be seen that

$$\mathbf{W} = -E\{(\mathbf{H}'\mathbf{H})\} \qquad \mathbf{I} = E\{(\mathbf{H}'\underline{\dot{x}})\} \qquad (26)$$

The secret of system parameter identification using the Hopfield model is revealed by Equation (23) to be the steady-state solution of the network. Further, the performance index gradient in Equation (24) is contained in the time derivative $d\Theta/dt$.

Though the association between the adaptation of the Hopfield network for system parameter identification and Hopfield's own proof of the minimization of a bounded energy function has been shown, it can not yet be said that stability for the new formulation has been proven. This proof is contained in Appendix A and is based on showing the minimization of the positive-definite function J. The proof shows that the time derivative of J is always negative and therefore J will always seek a minimum value.

It is now clear that to use the Hopfield model for system parameter identification, one must collect all of the state variables of the system response, as well as the input, and

use them to formulate the Hopfield weight matrix **W** and bias matrix **I**. Then, one need solve the system of first-order ODE's of Equation (25) for the vector **V**. The solution obtained for **V** will, by definition, be a vector containing the elements of the **A** and **B** matrices of the state-space form of the system, and system parameter identification will be achieved.

Figure 4 is a block diagram showing this process. The "Averaging System" is a routine which accepts the time history of the system input and its state-space responses and formulates matrices **W** and **I** averaged over a certain time interval. This process was developed as a result of knowledge gained from experimentation and is explained in greater detail in the next chapter. The bias matrix **I** is added to the negative product of the weight matrix **W** and the vector of previous neuron outputs **V**. This sum is multiplied by a scale factor "s1" to speed the response of the system and the result is a time derivative which, when integrated, yields a vector of the weighted summation of inputs from all other neurons. This vector is operated on by the non-linear activation function g to produce the output of the current neuron.

The following chapters in this thesis present the results of extensive experimentation with the theory of system parameter identification using the Hopfield network. Using a test case for a simple, time-invariant, second-order system where the response and the parameters are known, the Hopfield model is implemented in computer software to solve the

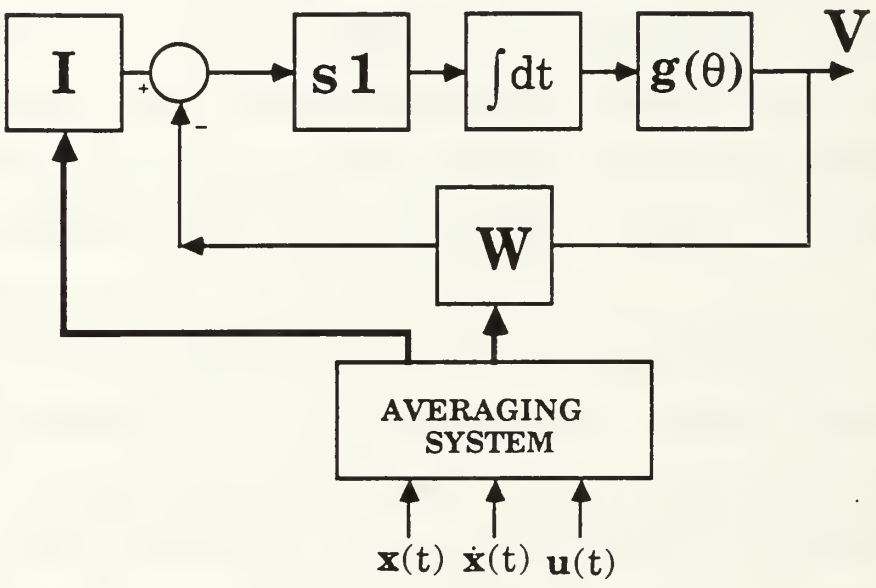

Figure 4. Block Diagram of Continuous Hopfield Network

characteristic system of first-order ODE's. Experiments are conducted using both linear and non-linear threshold functions g, and the issue of stability of the network is thoroughly explored. As an attempt at a more realistic scenario, the performance of the network is evaluated when the system response is not completely known, that is, not all state variables are measurable. Finally, in a case that would most certainly arise in the operating environment of NPS AUV II, parameter identification accuracy is evaluated when the parameters themselves change with time.

III. RESULTS OF THE HOPFIELD SIMULATION

A. INTRODUCTION

In this thesis, the theory of Shoureshi and Chu [Ref. 4] using the Hopfield algorithm for parameter identification has been applied to a known, second-order system herein referred to as the "test case." This test case is the same as that investigated in [Ref. 4] but the solutions have been found completely independently. The purpose of this exercise is to investigate the important aspects of the Hopfield algorithm as related to global stability, speed of convergence, most efficient formulation, and accuracy in identifying known parameters. This chapter provides a detailed analysis of this test case and results of the Hopfield formulations using both linear and non-linear transfer functions, the non-linear function being exclusively the so-called "sigmoid" function.

B. LINEAR TRANSFER FUNCTION

The test case system chosen is an oscillatory , second-order system with both states excited by the input signal, $\underline{u}(t)$. The matrices of constant parameters are

$$\mathbf{A} = \begin{bmatrix} -.94248 & 12.566 \\ -12.566 & -.94248 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1.0 \\ 2.0 \end{bmatrix} \tag{27}$$

The equations for the system in state-space form are written as follows:

$$\begin{Bmatrix} \dot{\underline{x}}_1 \\ \dot{\underline{x}}_2 \end{Bmatrix} = [\mathbf{A}] \begin{Bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{Bmatrix} + [\mathbf{B}]\{\underline{u}\} \tag{28}$$

where $\underline{u}=\sin(t)$ is the system input. This input function was chosen to produce persistent, system excitation. Figure 5 is a plot of the state variable response of this system along with the input $\underline{u}$ for initial conditions arbitrarily chosen as $x_o=[1\ 1]$.

Appendix B contains the computer code for this problem formulation, implemented using MATLAB software. The first step in the main program, called "neu," obtains an appropriate time history for the two state variables and their time derivatives from which the weight and bias matrices, **W** and **I**, are formed. This step is necessary because this approach to system identification requires measurement of $\underline{x}$, $\dot{\underline{x}}$, and $\underline{u}$. For this, a Runga-Kutta second- and third-order numerical integrator, provided by the MATLAB software, generates response data for the system of two, first-order, ordinary differential equations (ODE) making up the state-space formulation. The subroutine "system" contains the state-space equations where **A** is the dynamics matrix and **B** is the gain matrix. The MATLAB numerical integrator, a subroutine called "ode23," operates on these equations given the time interval specified by "to" and

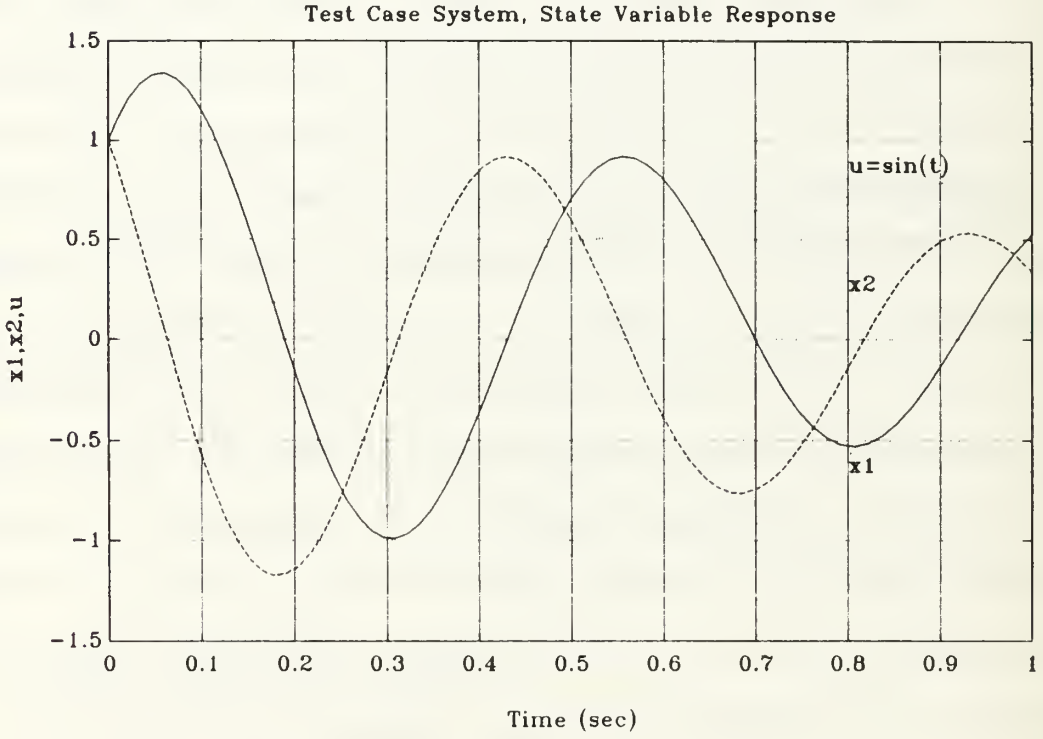

Figure 5. Test Case System, State Variable Response

"tf", and by the initial conditions "xo". For this formulation to=0.0, tf=1.0, and xo=[1 1]. Subroutine "ode23" chooses non-constant time intervals based on the speed of convergence of the solutions and returns the values of $\underline{t}$, $\underline{x}_1$, and $\underline{x}_2$.

A loop in the main program reintroduces these values for $\underline{t}$, $\underline{x}_1$, and $\underline{x}_2$ into the subroutine "system" to obtain the corresponding time derivatives of $\underline{x}_1$ and $\underline{x}_2$, designated $\underline{f}_1$ and $\underline{f}_2$. The input $\underline{u}$=sin(t) is calculated given the time steps provided by "ode23." Program "neu" formulates the matrices **W** and **I** from the time histories of $\underline{x}_1$, $\underline{x}_2$, $\underline{\dot{x}}_1$, $\underline{\dot{x}}_2$, and $\underline{u}$, as presented previously. As the main program was first written, **W** and **I** were formulated for each time step, but this quickly proved to be unworkable as **W** is a singular matrix for each time step. As such, $\mathbf{W}^{-1}$ does not exist, so the steady state solution

$$\mathbf{V}_{ss} = -\mathbf{W}^{-1}\mathbf{I}' \qquad (29)$$

likewise does not exist.

To investigate this problem further, **W** was formulated as the average for each time interval of state variable measurement and the upper singular values (usv) of the eigenvalue pairs were calculated and plotted. Figure 6 is a plot of the usv's of the eigenvalues of **W** versus the number of time steps used to formulate **W** before averaging the elements to obtain a final **W**. Each curve shows a pair of singular values (the matrix is symmetric) and it can be seen that until

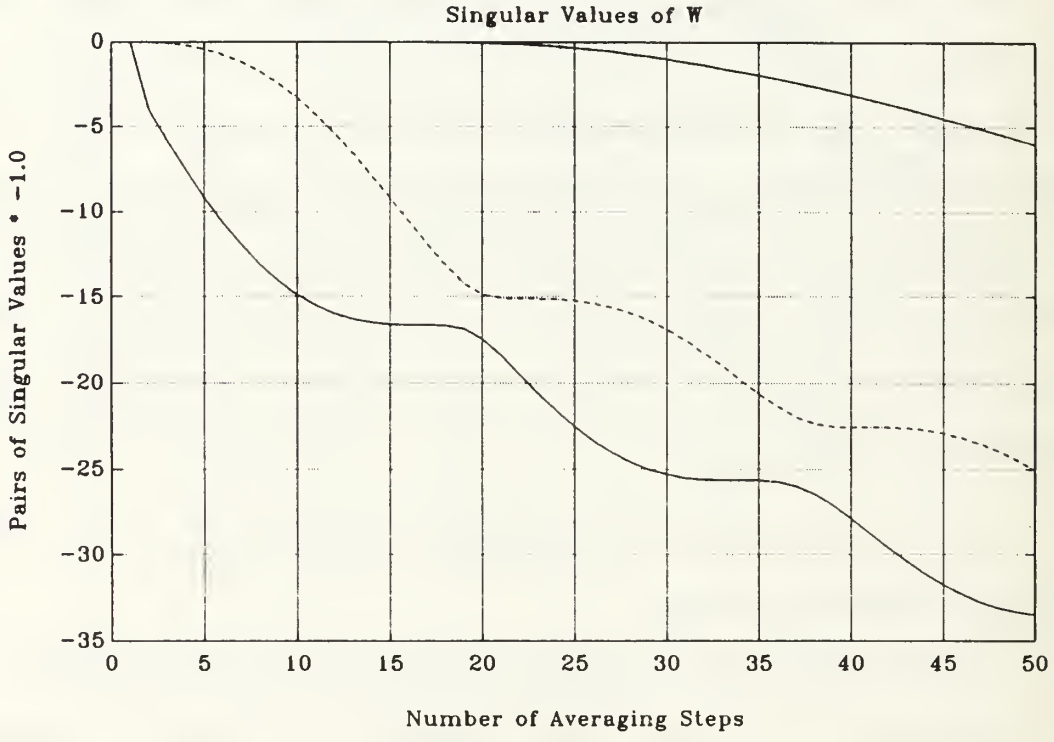

Figure 6. Singular Values of W

the number of averaging steps exceeds 30, one pair of eigenvalues is too close to zero to ensure the stability of the network solution. From this point on, 50 was taken as the appropriate number of time steps over which to average **W**. An optimization of this problem was not attempted but might be a subject of future research. To check that this was an acceptable formulation for **W**, the steady-state solution for the system parameters, Equation (29), was calculated for each time step of the state-space response. Figure 7 shows a plot of this steady state solution and it can be seen, as expected when the input signal is free of noise, that within three time steps the proper Θ_{ss} is returned. Note that in the figure, as in the computer code, Θ is denoted "th."

Once the proper number of averaging steps was determined, it was necessary to reset the vectors for $\underline{t}$ and $\underline{x}$ to contain only that number of elements. The main program, as currently written, cycles these re-sized vectors through the subroutine "system" to obtain their corresponding time derivatives in the matrix **f**. With this information the matrices **W** and **I** are formulated by multiplying these column vectors of $\underline{x}$, $\underline{f}$, and $\underline{u}$ as specified to produce each individual element, which in turn is divided by 50. What results is a 6X6 weight matrix **W** and a 6X1 bias matrix **I** which are averaged for the system state over roughly the first one second of time.

Subroutine "hop2" contains the Hopfield algorithm and comprises the next phase of the computer code. Program "neu"

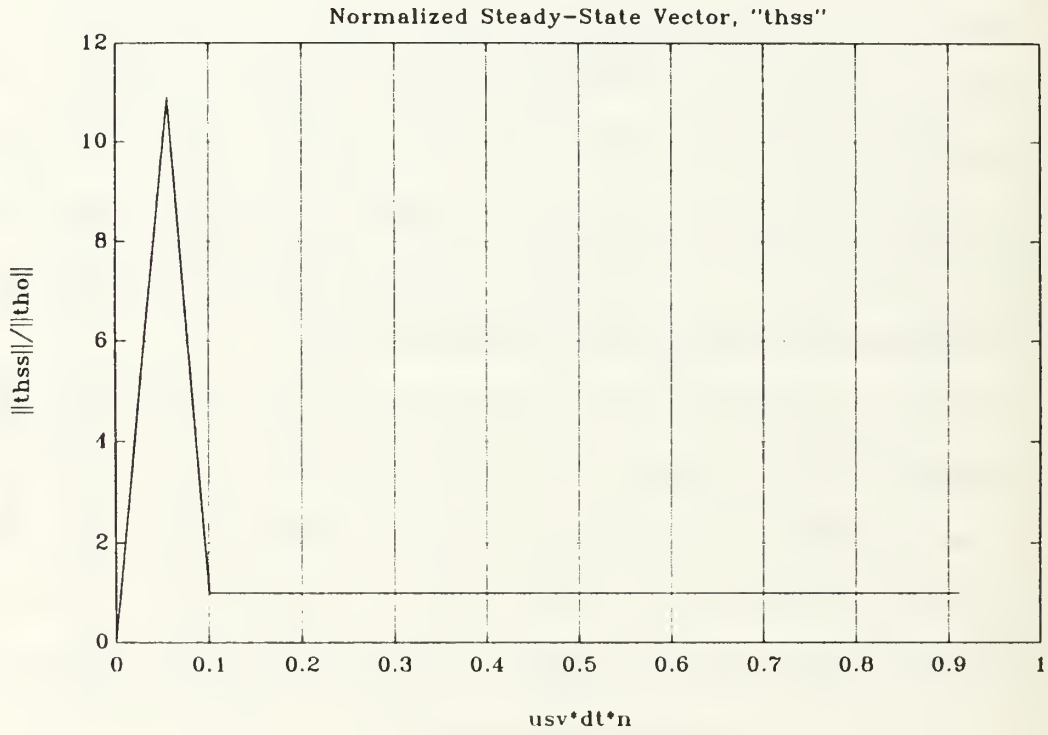

Figure 7. Normalized Steady-State Vector, "thss"

calls "hop2," entering with **W** and **I** and integrating the series of six, first-order, ODE's to produce a solution contained in the vector Θ. As before, the equations of the system are :

$$\frac{d\Theta_i}{dt} = -\sum_{i=1}^{6}\sum_{j=1}^{6} \mathbf{W}_{ij}\mathbf{V}_j + \mathbf{I}_i \qquad (30)$$

The linearity of this particular system lies in **V**, which for this first test case is represented by:

$$\mathbf{V} = \Theta \qquad (31)$$

A scale factor, s1=50, multiplies **W** and **I** to increase the response speed.

Program "neu" uses a simple Euler integration, where it was determined by experimentation that dt=.02 produced acceptable convergence of the solution to the final values of Θ provided integration was carried out over 100 steps. As will be discussed in the section on non-linear transfer functions, it was discovered that the actual, desired solution for the system parameters is returned in **V**, not Θ. This was overlooked at first because of the constraint imposed by Equation (31), which essentially means that there was a unit transfer function for the linear case.

Figure 8 shows a plot of each individual element of Θ normalized with respect to the actual system parameters versus the non-dimensionalized time of Euler integration. It can be seen that all elements converge to the expected values within $\omega_o t$=20. Figure 9 is a plot of the normalized Θ vector and

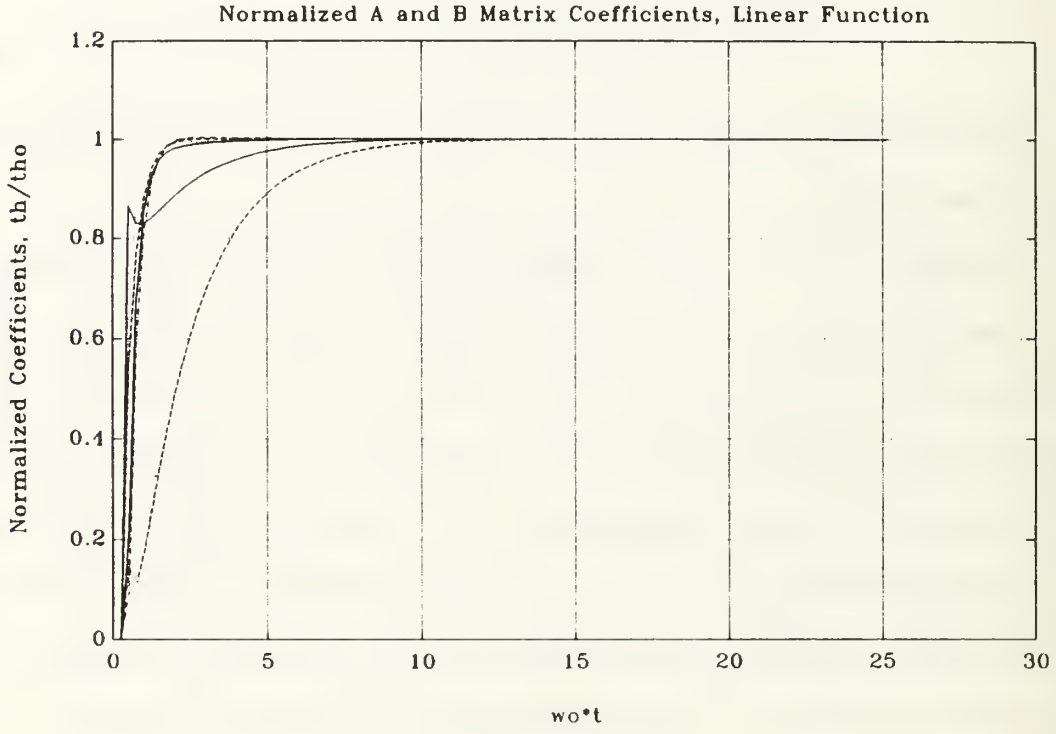

Figure 8. Normalized A and B Matrix Coefficients, Linear Function

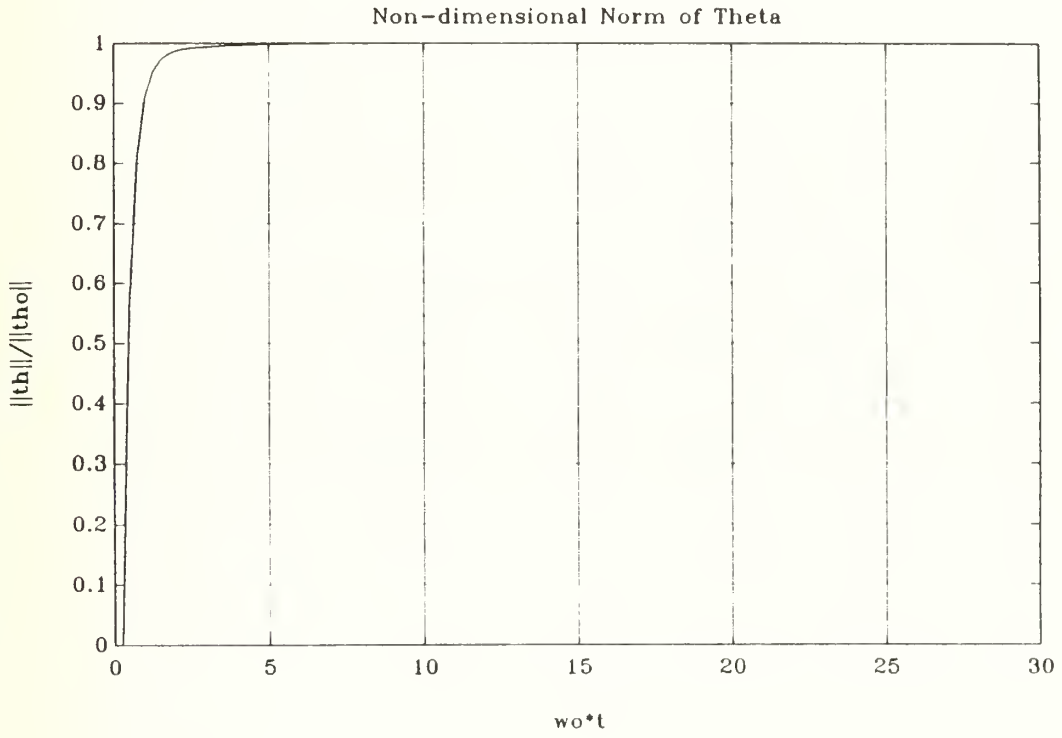

Figure 9. Non-Dimensional Norm of Theta

shows that convergence to the actual system parameter vector Θ_o occurs within $\omega_o t=15$.

C. NON-LINEAR, SIGMOID, TRANSFER FUNCTION

The case where the Hopfield network was implemented using a linear transfer function has been thoroughly investigated. It has proven useful for the purposes of exploring the important aspects of stability associated with this problem. In addition, the linearity of the system guarantees that a solution will be obtained provided **W** is not singular. The non-singularity of **W** is ensured by averaging over an appropriate time interval.

The network as described is not strictly a Hopfield network, however, because the linear transfer function does not accurately model the response of a neuron. In Hopfield's formulation [Ref. 6] the transfer function which converts the output of each neuron is more accurately represented by the non-linear sigmoid function in Equation (14) where λ is the learning rate and is positive but less than one. The plot of this sigmoid function, Figure 3, shows its relationship to the ideal step change from zero to one.

The sigmoid function is the basis for the next set of experiments which were conducted with the test case used previously. Initially, all parameters remained the same as for the linear network. Additional parameters that needed consideration were the learning rate, λ, and the sigmoid

function gain, G. Prior to conducting the experiments it was not immediately apparent that G would be necessary, but its eventual inclusion proved critical and will be discussed further.

Figures (10) and (11) are two plots of the sigmoid function which highlight the effect of varying λ and G. It should be noted that in order to duplicate, and closely follow, the test case used in [Ref. 4], the form of the sigmoid function was altered to

$$g(\Theta) = G\left[\left(\frac{2}{1 + e^{-\lambda\Theta}}\right) - 1\right] \qquad (32)$$

In Figure 10, G is held constant at 1 while λ is varied from 0.1 to 1.0. The function converges asymptotically to +1 and -1 for all λ, but the slope of the linear portion of the function increases dramatically with increasing λ. Figure 11 shows λ held constant at 0.1 while G varies from 10 to 50. The asymptotes vary directly with G with an equally dramatic increase in slope as G increases.

Initially, λ was set at 0.1. As the necessity of G was not yet recognized, it was not included, effectively making G equal to 1. Figure 12 shows the results of this run, revealing that each of the coefficients in the system output vector, **V**, converges within the range +1 to -1. Experiments with various values for λ, as well as expanding the time interval for

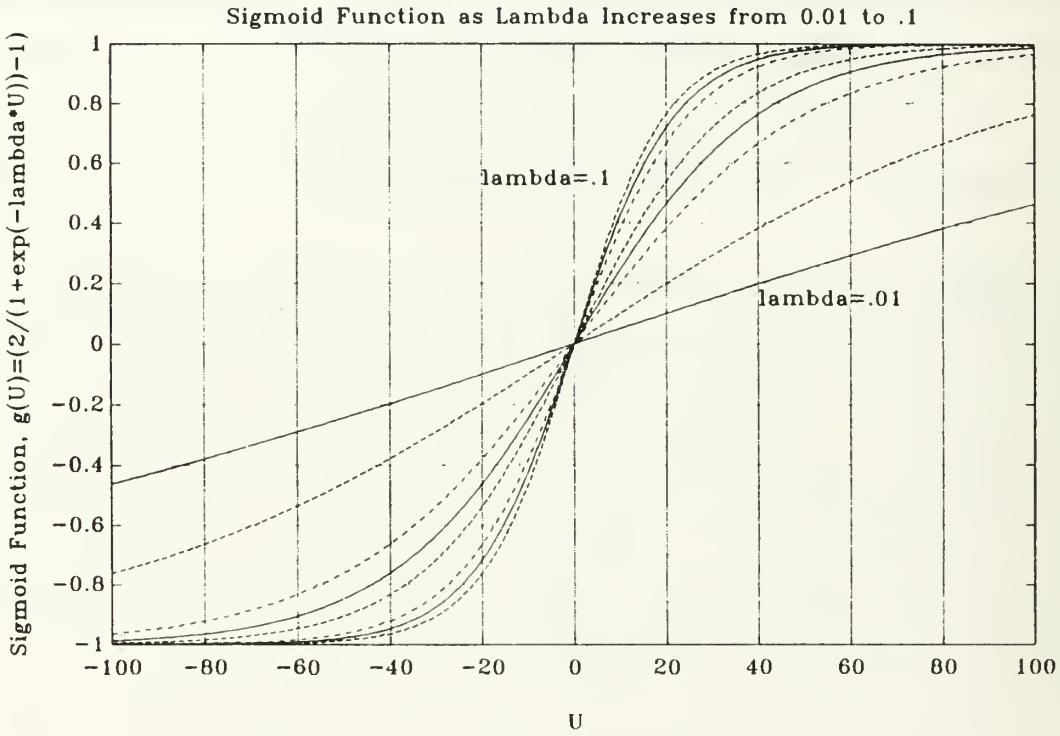

Figure 10. Sigmoid Function as λ Increases From 0.01 to 0.1

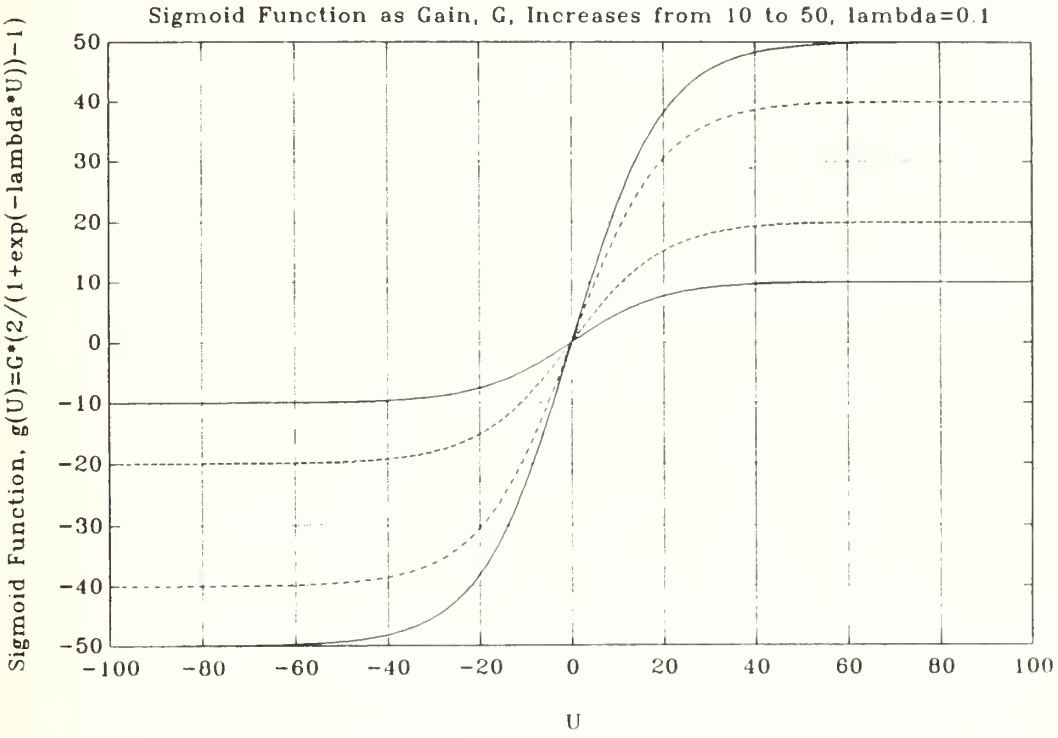

Figure 11. Sigmoid Function as Gain, G, Increases From 10 to 50, $\lambda = 0.1$

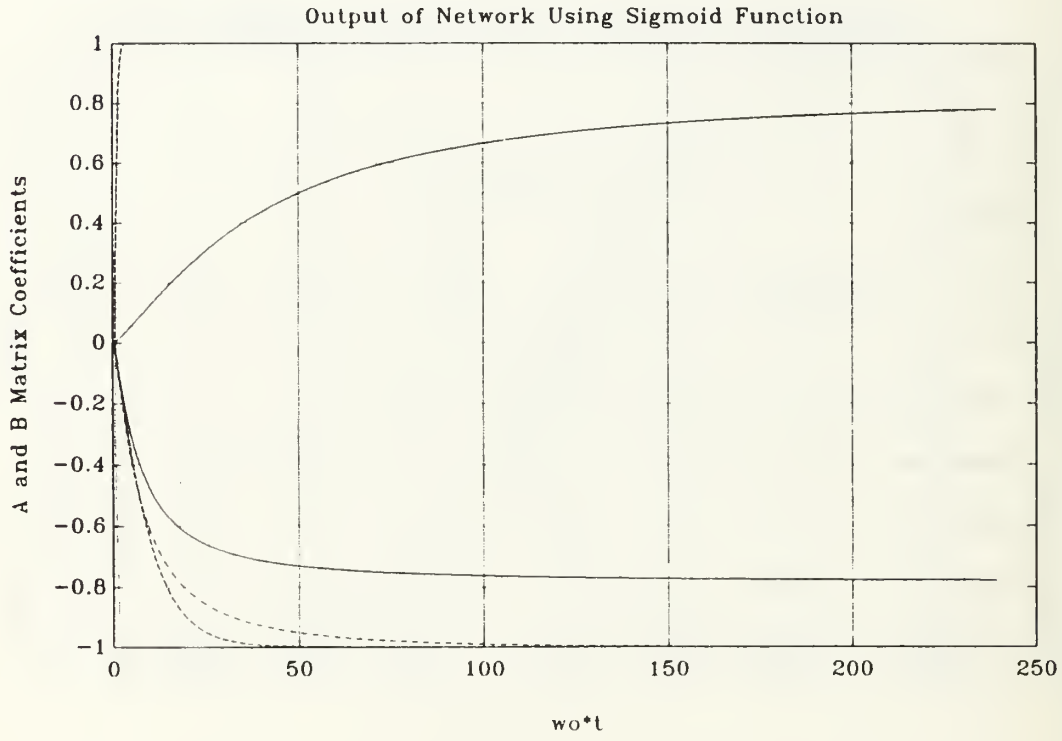

Figure 12. Output of Network Using Sigmoid Function

integrating the network, proved fruitless in changing this incorrect "solution."

The reason for this failure to converge to the expected results with what was believed to be Hopfield's original formulation became apparent once the network was analyzed mathematically with $g(\Theta)$ included. This system formulation is as follows:

$$\frac{d\Theta_i}{dt} = -\sum_{j=1}^{N} \mathbf{W}_{ij} g(\Theta_i) + \mathbf{I}_i \qquad (33)$$

As $t \rightarrow \infty$, $(d\Theta_i/dt) \rightarrow 0$, so

$$-\sum_{j=1}^{N} \mathbf{W}_{ij} g(\Theta_i) + \mathbf{I}_i = 0 \qquad (34)$$

and

$$g(\Theta_i) = \sum_{j=1}^{N} (\mathbf{W}_{ij}^{-1}) \mathbf{I}_i \qquad (35)$$

Note, however, that the right hand side of Equation (35) returns the steady state values of the system parameters while the left hand side by definition must vary between -1 and +1. Thus, the output $g(\Theta)$ is constrained to converge to these values. This response provided the clue that some gain contained within $g(\Theta)$, which would extend the range of the sigmoid function to encompass the expected values of the system parameters, **is required for the existence of a steady-state solution**.

When devising a method to determine the proper values of G and λ to use, it was decided to maintain the same unit value for the slope of the linear section of the transfer function. For unit slope, then, the derivative with respect to Θ of Equation (32) must equal one at Θ=0, or

$$\frac{\partial g(\Theta)}{\partial \Theta} = G[2(-1)(1+e^{-\lambda\Theta})^{-2}(-\lambda e^{-\lambda\Theta})] \qquad (36)$$

and

$$\frac{\partial g}{\partial \Theta}\Big|_{\Theta=0} = 1 = G[-2(2)^{-2}(-\lambda)] \qquad (37)$$

so

$$1 = G\left(\frac{\lambda}{2}\right) \qquad (38)$$

and

$$G = \frac{2}{\lambda} \qquad (39)$$

A value of G=15 was chosen to encompass the largest of the expected values of the system parameters, resulting in λ=0.133, so these values were incorporated into the Hopfield network. Figure 13 shows various plots of the sigmoid function when unit slope is maintained according to Equation (39).

Appendix C contains a listing of the new computer code incorporating the sigmoid transfer function for network formulation. The main program, called "neu2sig," calls a reformulated Hopfield network subroutine "hop2sig," and integrates the series of equations where **V** now equals $g(\Theta)$,

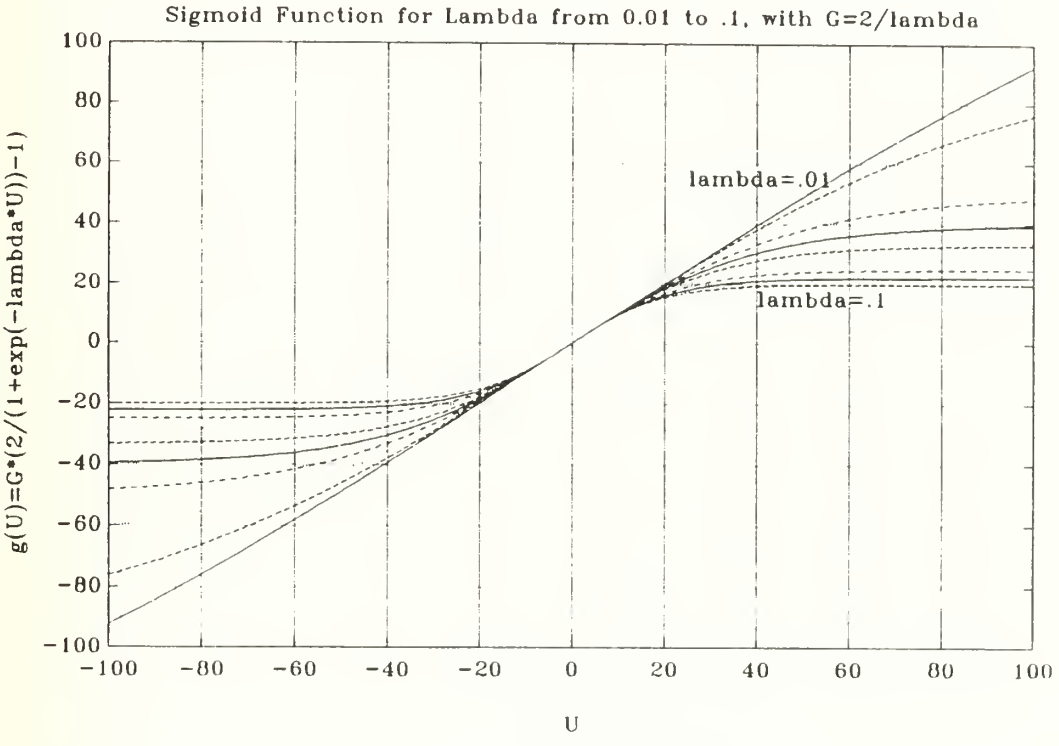

Figure 13. Sigmoid Function for λ From 0.01 to 0.1, with G = $2/\lambda$

the sigmoid function. As discussed previously, the system parameters are returned in **V**, not Θ, so a plot of the elements of **V**, as the Euler integration proceeds, reveals the behavior of the network as time advances. Figure 14 is a plot of the separate elements of **V** normalized with respect to the individual, expected values of the system parameters, **A** and **B**, showing their convergence within $\omega_o t=20$. Figure 15 is a similar plot, except that **V** is plotted as a vector normalized with respect to the vector of expected system parameters. This plot shows convergence to within $\omega_o t=15$.

D. REMARKS

It has been shown that the Hopfield network algorithm can be used to identify the parameters of a simple, time-invariant, second-order system provided that the expected values of the parameters lie within the range of values of the non-linear transfer function used to convert the output of each neuron. The application of this method to real-world system identification is limited, however, by the necessity of having all states of the system fully measurable. A more realistic, and therefore more important, situation is one in which at least one system state is not observable, meaning this state must be estimated before the system parameters may be identified. The next chapter details the formulation of a Hopfield network for the current second-order test case, with

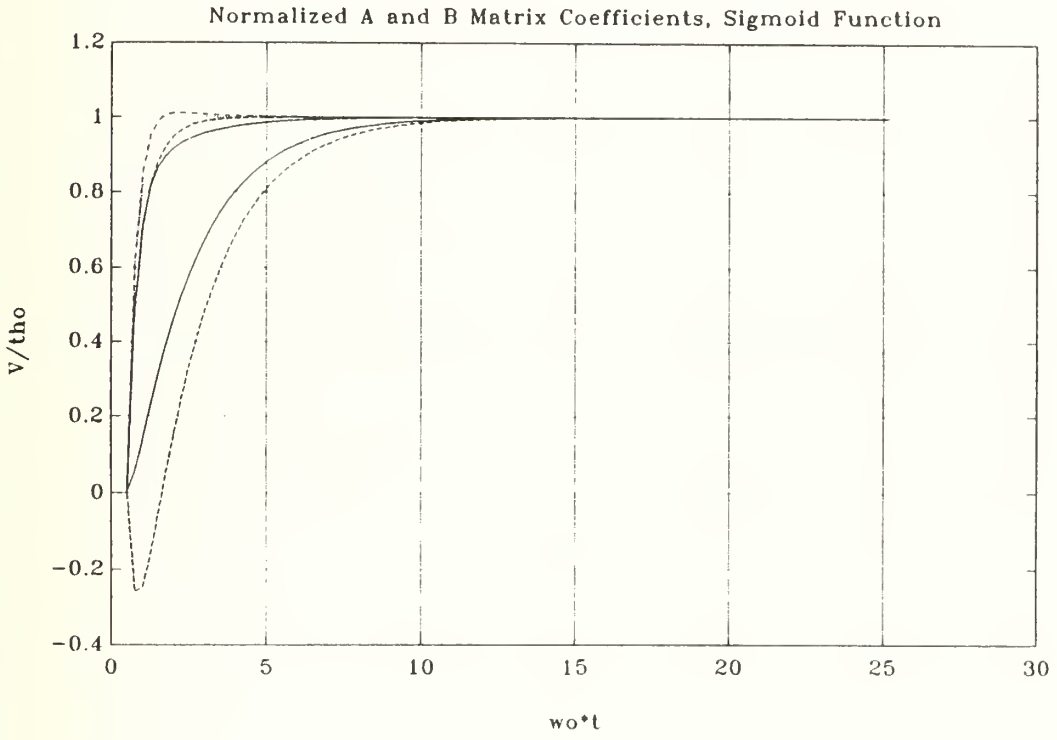

Figure 14. Normalized A and B Matrix Coefficients, Sigmoid Function

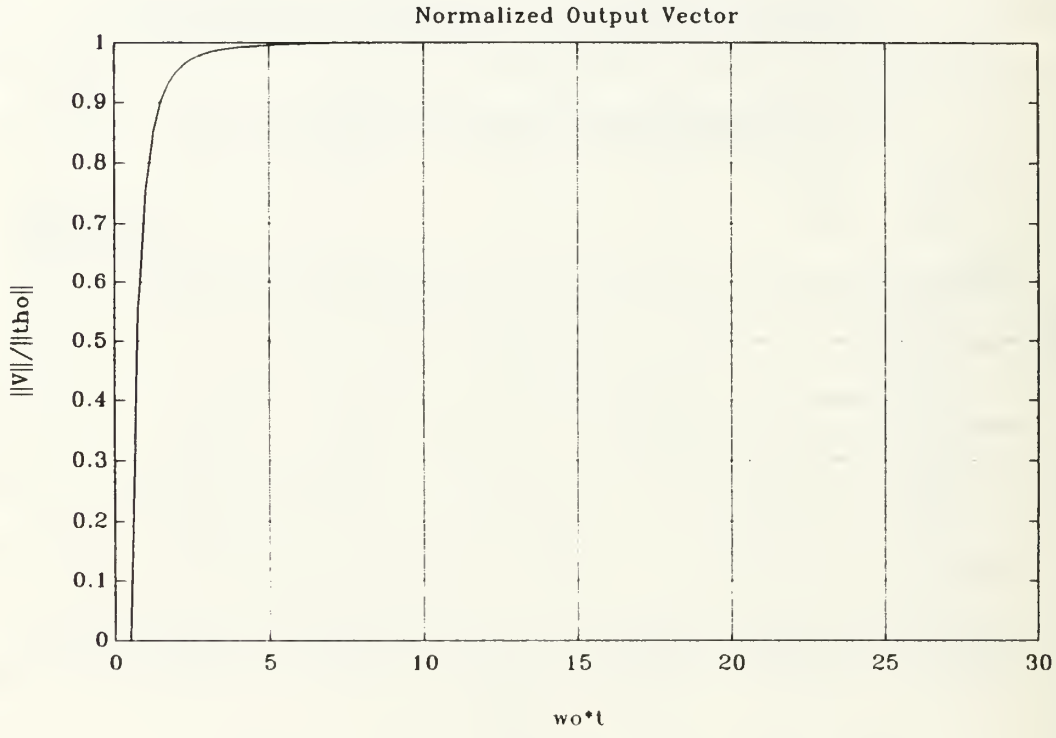

Figure 15. Normalized Output Vector

the added complication that the state denoted x is not measurable.

IV. PARAMETER IDENTIFICATION USING STATE OBSERVERS

A. INTRODUCTION

The need for observers in modern control system design where one or more states can not be measured has been thoroughly explored and documented. The original test case as previously used is now adapted to the exploration of state observation applied to the Hopfield network algorithm. Although the full states have been generated previously and are known, this Chapter examines the case where only the state variable $\underline{x}_2$ is measurable, while $\underline{x}_1$ must be estimated. This estimated variable, denoted $\underline{\hat{x}}_1$, will be used with an estimated variable $\underline{\hat{x}}_2$ to formulate **W** and **I**. Finally, it will be determined if the Hopfield network will correctly identify the original system parameters under these conditions.

B. FULL-STATE OBSERVER

For the exploration of this problem, the computer code developed originally was left largely intact. However, the subroutine "system," used previously to generate a time history of the state variables, has been replaced with a subroutine called "observer" which contains the code necessary to create a time history of the estimated variables $\underline{\hat{x}}_1$ and $\underline{\hat{x}}_2$. Appendix D contains the pertinent computer routines for this problem. Note that in the code $\underline{\hat{x}}_1$ and $\underline{\hat{x}}_2$ are contained in the

matrix "xhat," while the time derivatives $\dot{\underline{\hat{x}}}_1$ and $\dot{\underline{\hat{x}}}_2$ are contained in "xhatdot."

Subroutine "observer" now contains a system of four first-order ODE's of the following form:

$$\begin{Bmatrix} \dot{\underline{x}}_1 \\ \dot{\underline{x}}_2 \end{Bmatrix} = [\mathbf{A}] \begin{Bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{Bmatrix} + [\mathbf{B}]\{\underline{u}\}$$

$$\begin{Bmatrix} \dot{\underline{\hat{x}}}_1 \\ \dot{\underline{\hat{x}}}_2 \end{Bmatrix} = [\mathbf{A}_o - \mathbf{K}\mathbf{C}_o] \begin{Bmatrix} \underline{\hat{x}}_1 \\ \underline{\hat{x}}_2 \end{Bmatrix} + [\mathbf{B}_o]\{\underline{u}\} + [\mathbf{K}\mathbf{C}_o] \begin{Bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{Bmatrix} \qquad (40)$$

The matrices $\mathbf{A}_o$ and $\mathbf{B}_o$ are the observer system parameters which initially are set as exactly equal to the actual system parameters **A** and **B** in Equation (27). The matrix $\mathbf{C}_o$ is the output matrix, which in this case is [0 1]. The purpose of this "perfect estimation" of the actual system parameters is to see if the Hopfield network can correctly identify the actual system parameters based on state variable data produced by estimation. System identification through the use of an observer when the actual system parameters can not be accurately determined will be explored later in this chapter.

The feedback gain matrix **K**, which is the key to minimizing the error between the actual and estimated states, is found using the dual form of the linear quadratic regulator subroutine "lqr" provided by the MATLAB software. This subroutine returns the optimal value of **K** such that the feedback law

$$u = -\mathbf{K}x \tag{41}$$

minimizes the cost function

$$J = \int (x'Qx + u'Ru)\,dt \tag{42}$$

For purposes of this solution Q was chosen to be an identity matrix and the control weight R=.01, constrained by the state equation

$$\dot{\underline{x}} = \mathbf{A}\underline{x} + \mathbf{B}\underline{u} \tag{43}$$

The main program for this new problem formulation is essentially the same as for the fully measurable system but was renamed "neuobs." For the sigmoid function λ=0.1 and G=20, which were found to give slightly faster convergence of the solution to the expected values. The major difference in this main program is that **W** and **I** are calculated using $\underline{\hat{x}}_1$, $\underline{\hat{x}}_2$, $\dot{\underline{\hat{x}}}_1$, $\dot{\underline{\hat{x}}}_2$. As before, the final solution is returned in the elements of the vector **V**.

Figure 16 shows the convergence of the elements of **V** to the expected values of the actual system parameters. The speed of convergence is essentially unchanged from that of the previous system where all states were measurable, as shown by the convergence of all within the non-dimensional time value $\omega_o t$=20. Figure 17 shows the normalized vector ||**V**||; again, the convergence rate is very close to that of the fully measurable system.

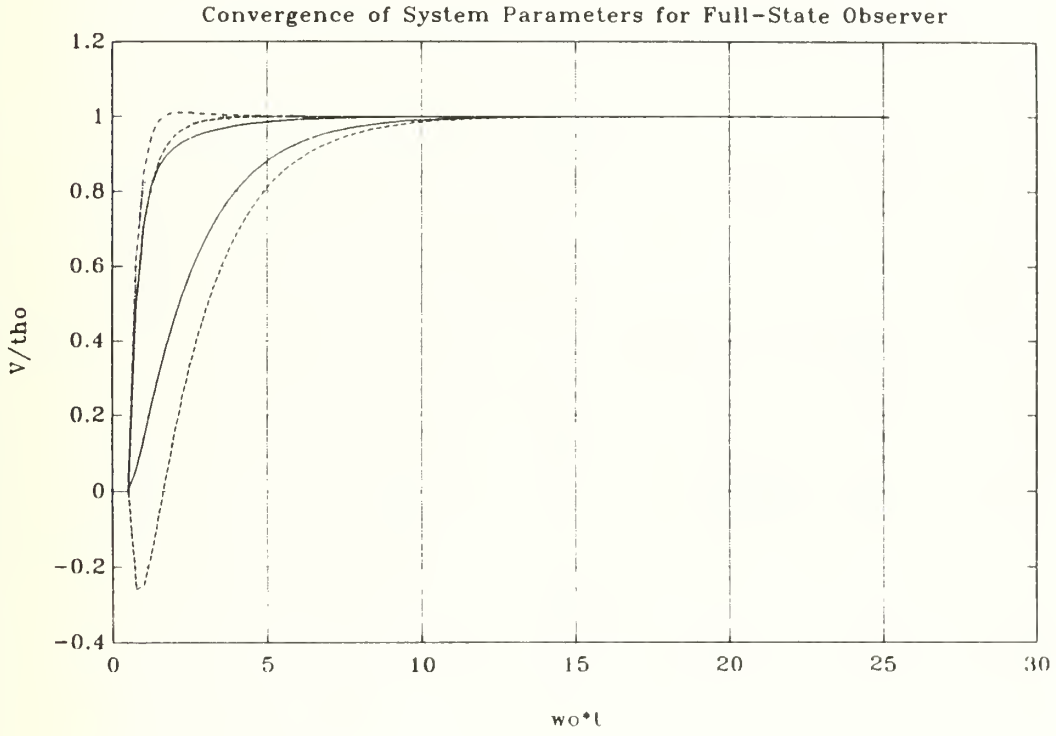

Figure 16. Convergence of System Parameters for Full-State Observer

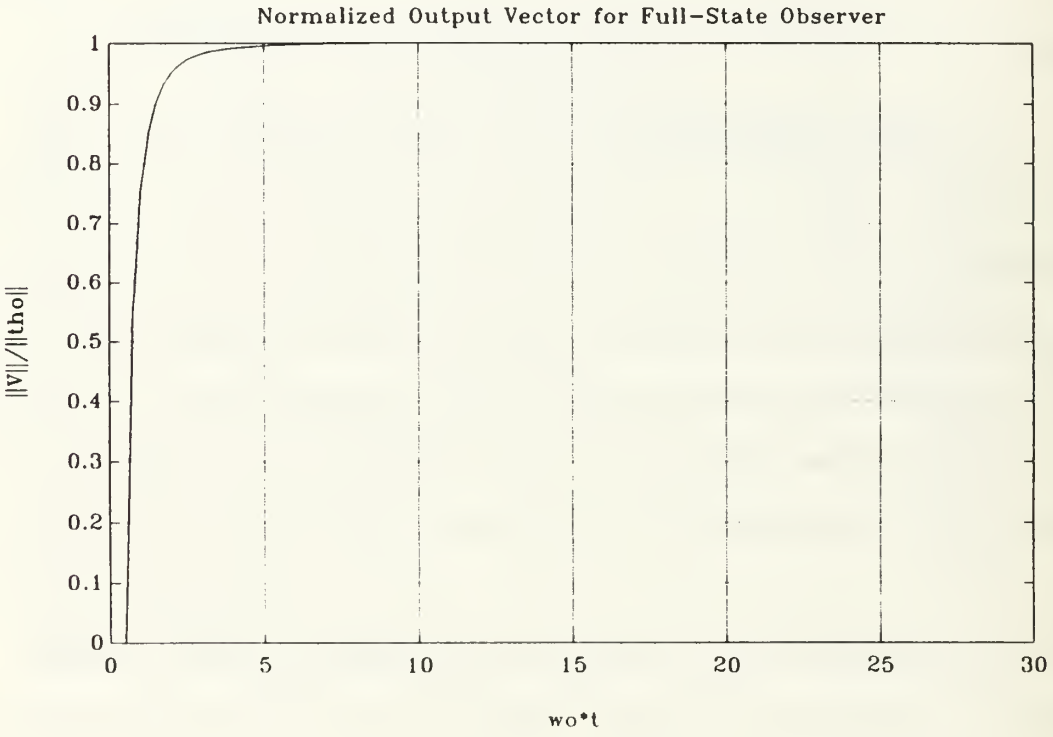

Figure 17. Normalized Output Vector for Full-State Observer

The next step in exploring the observer problem was to determine if the parameters of the system could be successfully identified for that case where the actual system parameters were not accurately known. For this phase of the problem the following observer matrices were chosen as initial estimates which differ substantially from the actual parameters of the test case:

$$\mathbf{A}_o = \begin{bmatrix} -.7 & 10.0 \\ -10.0 & -.7 \end{bmatrix} \quad \mathbf{B}_o = \begin{bmatrix} 2.44 \\ 1.39 \end{bmatrix} \tag{44}$$

Figure 18 is a plot of the resulting solution vector, **V**, where each element has been normalized over the expected value of its corresponding system parameter, and it shows that the network does not converge to the expected values of the actual system parameters. Table II shows the actual values of the system parameters, the values of the imperfectly estimated system parameters, and the steady-state solution of the network using the observer. It can be seen that the steady-state of the network returns something close to the value of the estimated parameter for $\mathbf{V}_{ss}(3)$ but all other values differ substantially from either the estimated or actual system parameters. Since this data is inconclusive, Figure 19 was generated which is a plot of the actual and estimated system state responses. This plot shows that the estimation of x_1 and x_2 begins to deteriorate rapidly at about t=0.7 and becomes so severe that it may indicate the presence of instability in the

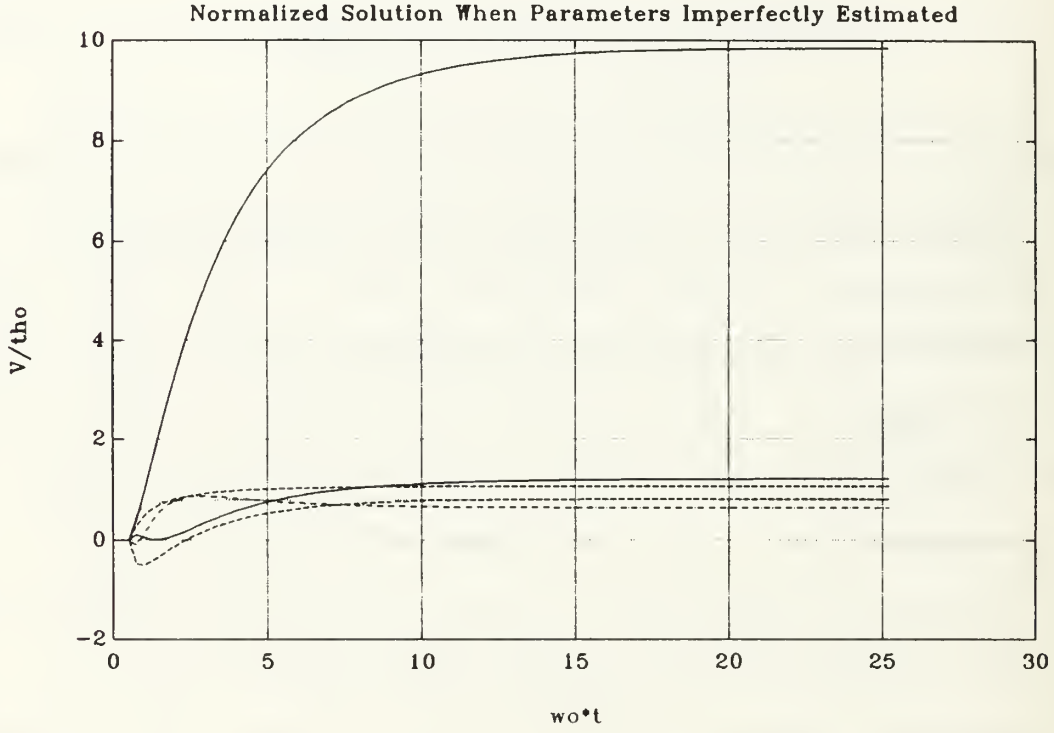

Figure 18. Normalized Solution When Parameters Imperfectly Estimated, Full State

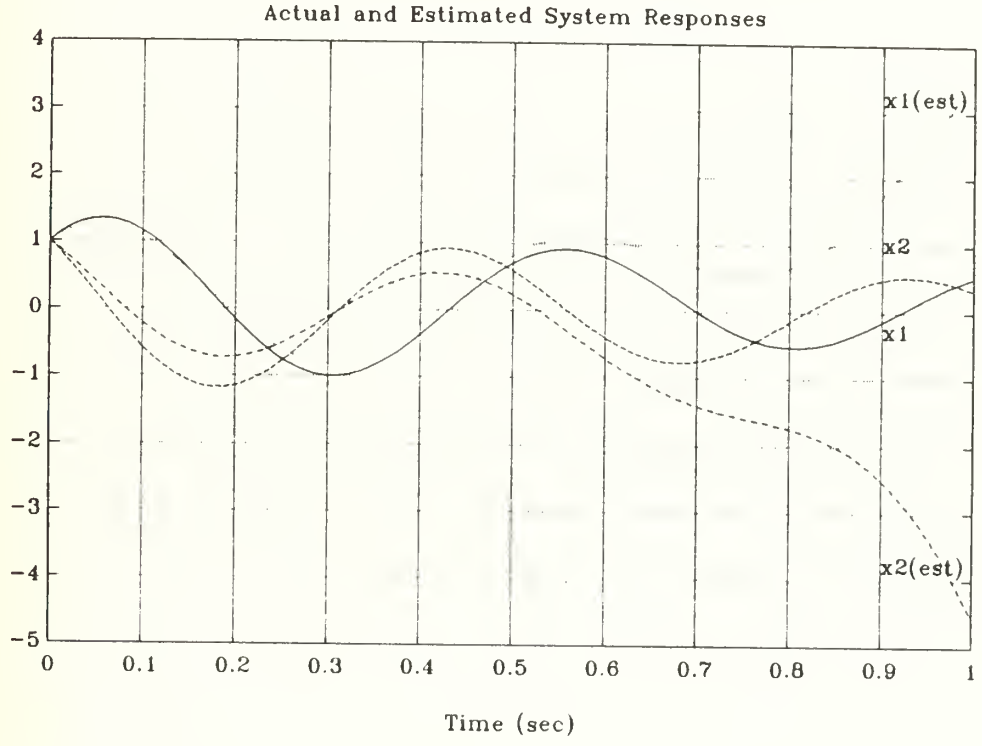

Figure 19. Actual and Estimated System Response, Full State

TABLE II. COMPARISON BETWEEN ACTUAL SYSTEM PARAMETERS AND SOLUTION USING FULL-STATE OBSERVER

A, B	A_o, B_o	V_{ss}
- 0.9425	- 0.7000	- 1.1368
12.5660	10.0000	13.3263
-12.5660	-10.0000	-10.0132
- 0.9425	- 0.7000	- 0.5991
1.0000	2.4400	9.8516
2.0000	1.3900	1.6147

observer. The inability of this system to produce an accurate estimation of the system response, which leads to inaccurate identification of system parameters, reveals the need for additional study into a method for minimizing the estimation error of the system response before formulation of the weight and bias matrices. Such a study was not conducted for this thesis but is deemed essential before implementation of a Hopfield network in a real-world application.

C. REDUCED-ORDER OBSERVER

The previous section dealt with the full-state observer applied to the Hopfield network, where the estimation of the entire state was returned if even one state variable in a multi-variable system was not measurable. As noted in Friedland [Ref. 14], however, it was desirable to formulate an observer which need only return estimation of those state variables not actually measured while omitting the need to

estimate variables already known. A reduced-order observer was used for this purpose of creating a time history of the state variables for the Hopfield network test case.

Friedland presents two methods of formulating the reduced-order observer depending on the nature of the eigenvalues of the submatrix $\mathbf{A}_{22}$. For the first method, the eigenvalues of $\mathbf{A}_{22}$ must be known to be negative, or that the real parts of the poles of this submatrix lie in the left half-plane sufficiently far from zero to ensure stability of the system. In this case, the equation to estimate the unmeasured variables is quite straight-forward and takes the form

$$\dot{\underline{\hat{x}}} = \mathbf{A}_{21}\mathbf{C}^{-1}\underline{y} + \mathbf{A}_{22}\underline{\hat{x}}_2 + \mathbf{B}_2\underline{u}. \tag{45}$$

However, when the eigenvalues of $\mathbf{A}_{22}$ are not known or if $\mathbf{A}_{22}$ is not stable, Friedland presents a more general method to estimate the unmeasured states. This second method was not needed for the test case examined in this thesis and it is not anticipated to be necessary for application to NPS AUV II, so its formulation has not been developed here.

Appendix E shows the computer code for the chosen observer method. The modified subroutine "observer" contains a system of three first-order ODE's consisting of Equations (28) and (45) adapted to estimate $\underline{\hat{x}}_1$. The main program "neuobs" selects $\underline{x}_2$ and $\underline{\hat{x}}_1$, retrieves $\underline{\dot{x}}_2$ and $\dot{\underline{\hat{x}}}_1$ from the subroutine "observer," and uses them in formulating **W** and **I**. From there the network

algorithm continues as before. Figure 20 shows the individual elements of **V** as they converge to the expected values of the system parameters with a speed comparable to that of the fully measurable system.

As with the full-state observer, experiments were conducted where the system parameters were imperfectly estimated as being those from Equation (44). Figure 21 shows that once again, the network solution converges to values substantially different from the actual system parameters as shown by the comparison in Table III. For the situation where x_2 is measurable but x_1 is estimated, the steady-state solution of the network converges to the imperfectly estimated parameter values $\mathbf{A}_o(1,1)$, $\mathbf{A}_o(1,2)$, and $\mathbf{B}_o(1)$. For the other three solution vector elements, convergence to some values close to neither the actual nor estimated parameter values has

TABLE III. COMPARISON BETWEEN ACTUAL PARAMETERS AND SOLUTION FOR REDUCED-ORDER OBSERVER

A, B	**A_o, B_o**	**V_{ss}**
- 0.9425	- 0.7000	- 0.7000
12.5660	10.0000	10.0000
-12.5660	-10.0000	-14.5529
- 0.9425	- 0.7000	- 1.5149
1.0000	2.4400	2.4400
2.0000	1.3900	12.6629

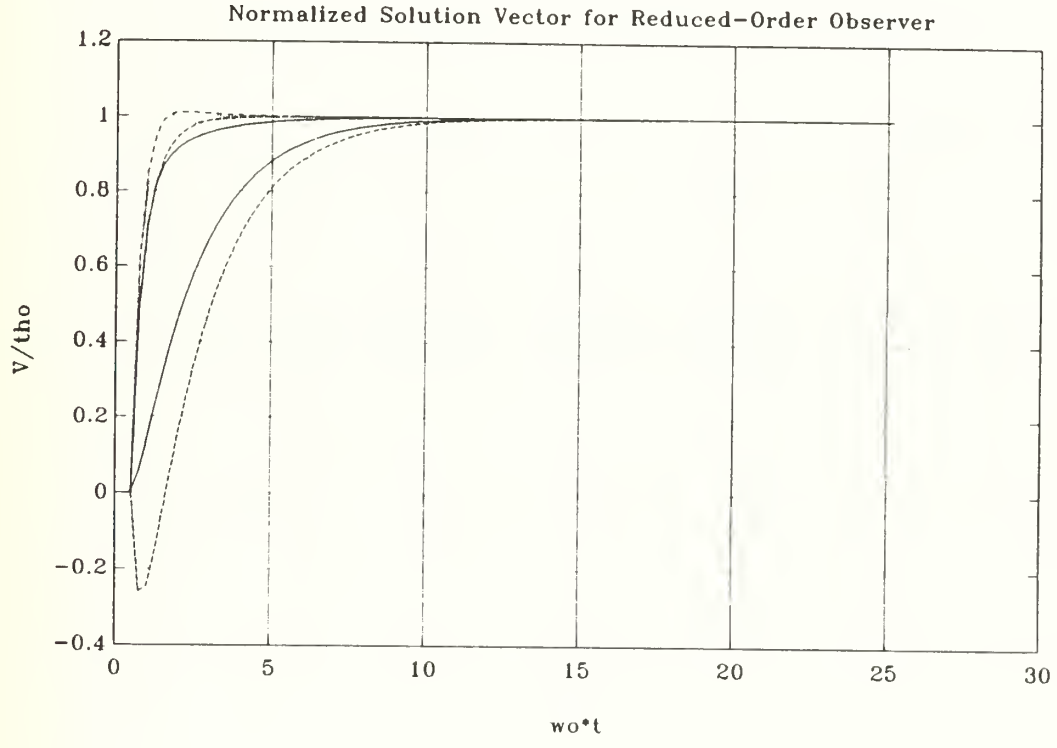

Figure 20. Normalized Solution Vector for Reduced-Order Observer

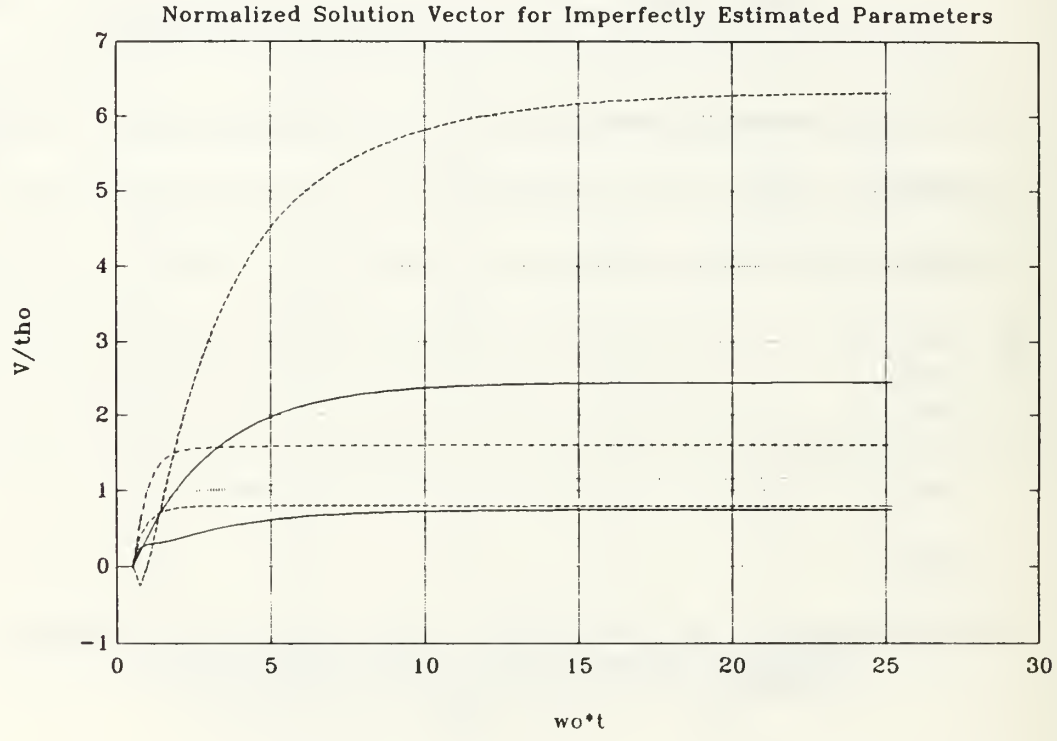

Figure 21. Normalized Solution Vector for Imperfectly Estimated Parameters, Reduced Order

resulted. Figure 22 is a plot of the system response for the observer network and it shows that the estimation of x_3 is not very accurate following t=0.2. Again, this may indicate instability in the observer system and, at least, requires an additional algorithm to minimize the estimation error before formulating **W** and **I**.

D. REMARKS

In this chapter the use of full-state and reduced-order observers has been explored for use with the Hopfield network for cases where the states of a system can not be fully measured and, therefore, the weight and bias matrices, **W** and **I**, can not be formulated based on actual system states. The need for a means of estimating certain state responses before implementing the control laws for the NPS AUV has become apparent early in it's design. When the surface dynamics of the vehicle are considered, it is found that two of the necessary system states, yaw rate r and heading angle ψ, can be easily measured with onboard sensors. However, the more subtle system state called side slip v, is not so easily measured. For this situation it is desired to adapt an observer which will estimate v with an acceptable degree of accuracy. As shown in this report, however, an accurate observer can not be developed unless there is full knowledge of the vehicle's system parameters, which are not now known and will most certainly be variable in the course of vehicle

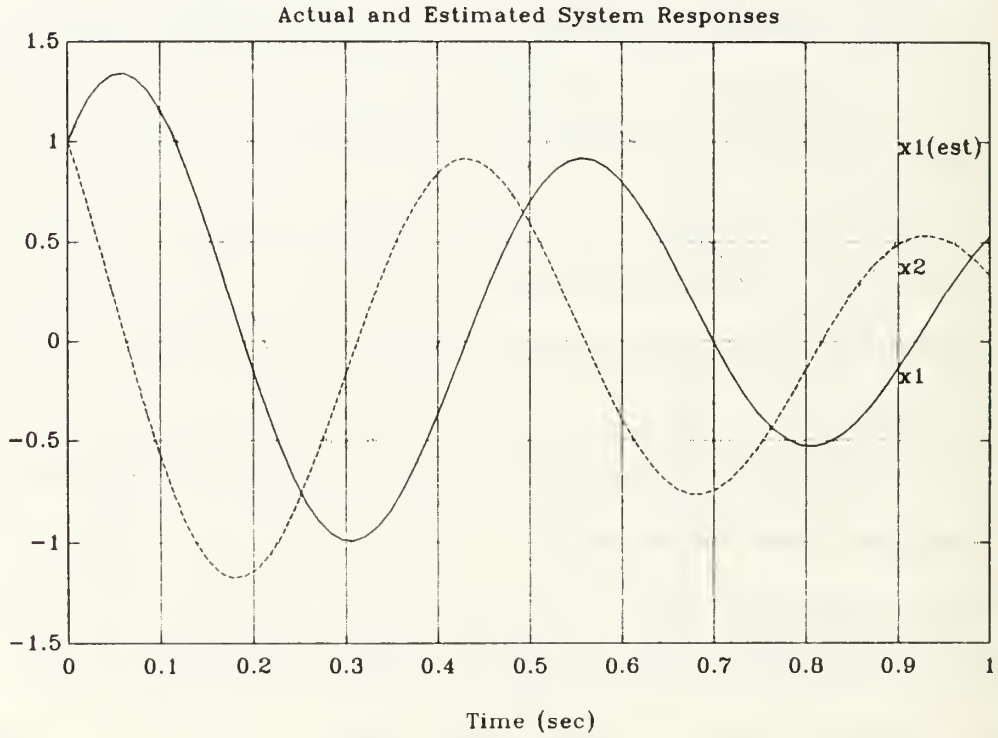

Figure 22. Actual and Estimated Responses, Reduced Order

operation. Since the purpose of this adaptation of the Hopfield network is for identification of the system parameters, clearly the use of an observer will be problematic.

V. TRACKING OF VARYING SYSTEM PARAMETERS

A. INTRODUCTION

An important situation to pursue in the study of system parameter identification is that of tracking and correctly identifying system parameters as they vary during system operation. Applied to the case of the NPS AUV II, such a situation may arise when one of the vehicle's control surfaces is lost or damaged. In this case, the dynamics of the vehicle would certainly change and, in order for the vehicle to continue to operate, the mission controller would have to compensate for these alterations and continue to function as specified. To do so, however, it must have the capability to accurately track the parameter variations in a timely manner.

In this chapter, the ability of the Hopfield network to track and identify such varying parameters is explored. For experimentation the test case in modified form is used where all states are assumed to be fully measurable. Simple step changes are imposed on one of the gain parameters, in this case [**B**(1)], and the results are plotted and examined.

B. RESULTS

Appendix F contains the code for tracking this system parameter. Subroutine "system" shows step changes implemented as simple "IF-THEN" statements, where [**B**(1)] varies once each

second from t=0.0 to 5.0. As before, the system response in terms of its state variables is produced using the MATLAB integration subroutine "ode23."

The main program "neuobs1" divides the time history into one-second intervals corresponding to the step changes, then reads the state variable values over those intervals. The state variables are used to formulate intermediate **W** and **I** matrices, averaged over the number of time steps within each interval, and the Hopfield network's system of ODE's is integrated using subroutine "hopobs" to produce a network solution corresponding to that interval of time. Once all intermediate network solutions have been generated, they are assembled into a global solution covering the entire five-second time period. Figure 23 shows a plot of [**V**(5)], which is the element of the network solution which tracks the parameter of interest. Overlaid on the plot is the actual variation of [**B**(1)], which allows the speed and accuracy of the convergence of the solution to be examined. The figure shows that for the first three seconds of time, the solution tracks the parameter values after converging within $\omega_o t=50$. This speed of convergence is much slower than for the case when the parameters remain constant, which for all experiments was on the order of $\omega_o t=20$.

Most importantly for this network solution, it can be seen from the figure that during the fourth, one-second time interval the error in the tracking element is unacceptably

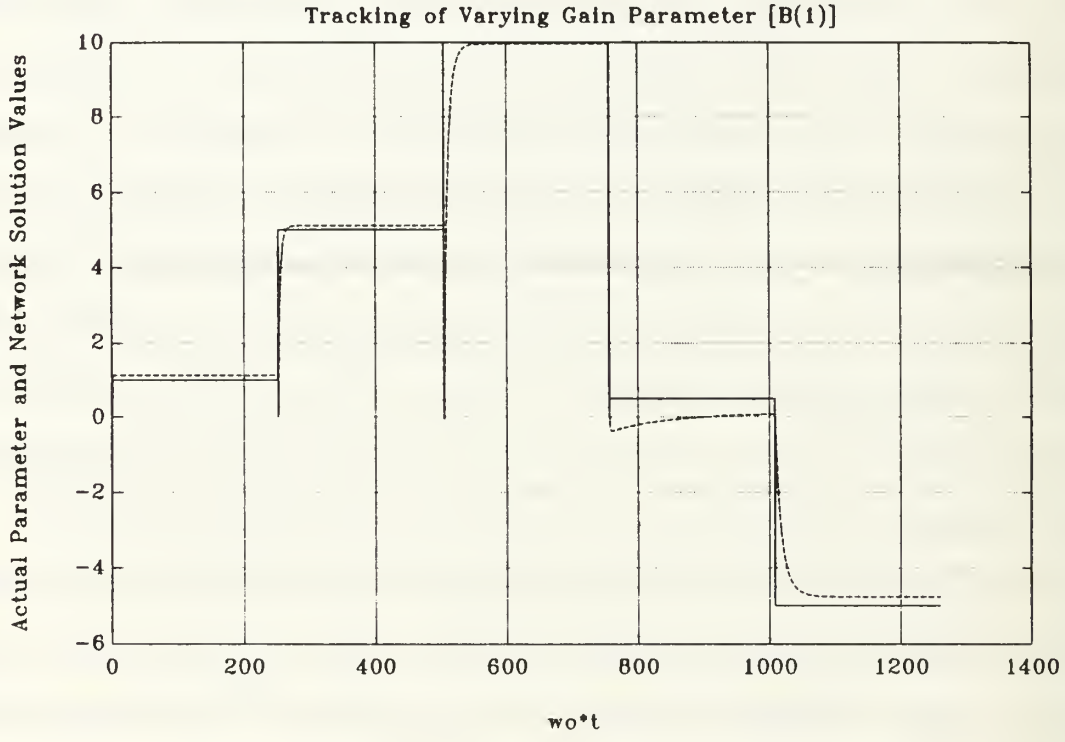

Figure 23. Tracking of Varying Gain Parameter [B(1)]

large and, in fact, the network has not achieved convergence to any value within $\omega_o t=200$. Following this discrepancy, during the fifth time interval, the speed of convergence and accuracy of the solution return to that noted for the first three time intervals. The behavior of the solution during the fourth time interval presents a serious lapse in the performance of the network and must be studied more closely.

The data for this network solution is presented in Tables IV and V in order that the performance of the network during the fourth time interval may be more closely studied. The eigenvalues of the weight matrix **W** shown in Table IV reveal that for the fourth time interval, two pairs of poles are extremely close to zero relative to the other pair and to the pairs of poles for the other time intervals. In addition, the small error found in the solutions for the other time intervals is also a product of lack of sufficient excitation. For the parameter tracking problem it was difficult to ensure that **W** would be averaged over the correct number of time steps because the subroutine "ode23" is a variable-time-interval integrator. This problem was not fully explored but should be a subject to pursue in future research involving parameter tracking.

Table V shows that the steady-state solution for the network element [**V**(5)] in the fourth time interval is equal to 0.1458 which is not at all close to the expected value for [**B**(1)] of 0.5. The reason behind this isolated discrepancy in

TABLE IV. EIGENVALUES OF W WHILE TRACKING [B(1)]

TIME INTERVAL	EIGENVALUE PAIRS		
0.0 - 1.0	.4024	.5014	.2679
1.0 - 2.0	.1347	.0902	1.1047
2.0 - 3.0	.0977	.0475	.7208
3.0 - 4.0	**.0035**	**.0025**	.2478
4.0 - 5.0	.0279	.0291	1.0183

TABLE V. STEADY STATE VALUES OF V ($V_{ss} = W^{-1}*I'$)

-.9044	12.5960	-12.5660	-.9425	1.1319	2.0000
-.9120	12.7052	-12.5660	-.9425	5.1165	2.0000
-.9310	12.5098	-12.5660	-.9425	9.9439	2.0000
-.5725	13.5320	-12.5660	-.9425	**0.1458**	2.0000
-.7734	12.1978	-12.5660	-.9425	-4.7667	2.0000

network performance lies in the fact that during the time interval in question, the gain parameter being tracked lies close to zero. During this period, the system is not being persistently excited to the extent necessary for the formulation of an appropriate **W**, which instead tends toward singularity. In this case, the network can not be expected to converge to the proper solution. An attempt was made to examine an alternate method of integrating the system to produce the state response. Instead of the variable-time-interval Runga-Kutta algorithm of "ode23" the system was integrated using a linear, time-invariant, time-response

algorithm with the system converted to discrete form. This method did not produce results substantially different than those shown in Tables IV and V.

It can be seen that in order for this network formulation to be used successfully under conditions of varying system parameters some adjustment in the software must be designed to guard against insufficient excitation by the system. It is apparent that test signals need to be defined to ensure suitable conditioning of **W**. A set of persistently exciting input functions could be stored which would be designed to maximize the singular values of **W**, which could be called by the vehicle's mission planner in order to optimally "test" the vehicle's behavior.

VI. SUMMARY AND CONCLUSION

A. SUMMARY

At the center of the Naval Postgraduate School's project to produce an Autonomous Underwater Vehicle (NPS AUV II) is the Mission Planning Expert System (MPES), a hierarchical system to control all operations of the vehicle while executing its planned mission. An important part of implementing such an expert system is mapping and successfully incorporating knowledge of the vehicle dynamics into a comprehensive vehicle control algorithm. One way of achieving this goal is through the application of Artificial Neural Networks (ANN) to identification of the state-space form of the vehicle's system parameters.

This thesis has focused on the application of the continuous form of the Hopfield network to system parameter identification. In its original form [Ref. 5], Hopfield presented his network as a discrete-time model of a set of biological neurons, each of which computes the weighted sum of the outputs of all other neurons, including its own, then sets the output to zero or one depending on whether it is above or below a set threshold value. In a subsequent paper [Ref. 6], he presented a continuous form of the model based on the operation of an electrical RC circuit. In that model, the

relation between the output of a neuron, termed its state, and the weighted sum of its inputs plus a bias value, was characterized by the non-linear sigmoid function, which he felt bears a superficial resemblance to the manner in which a biological neuron changes states.

For both models Hopfield sought to prove the network would always converge to a unique set of outputs based on a distinct pattern of inputs. To do this he likened the network to a bounded energy function which, as each neuron changes state, is monotonically decreasing and eventually seeks a minimum value.

Shoureshi and Chu used this idea of the minimization of an energy function while studying the problem of dynamic system parameter identification. Their work forms the basis of this thesis, which is a detailed study of the Hopfield network for parameter identification with a view toward implementing it into the control system of NPS AUV II. A linear, time-invariant system was chosen as a test case to see if the outputs of a Hopfield network would converge to the known system parameters, given that the weight and bias matrices, **W** and **I**, are formulated from the system response as shown herein. As the stability of the network in this form was also of major concern and not directly inferable from the Hopfield stability proof, a proof of stability was completed and is contained in Appendix A. Numerical experiments were conducted first with the system states fully measurable, then with the

incorporation of observers to estimate one state variable, and finally assuming the states were fully measurable but one parameter was varying with time.

B. CONCLUSION

It has been show that

1. Hopfield networks can solve function minimization quickly in real time and, in particular, can perform system parameter identification.

2. The formulation of the weight matrix **W** was not possible for each individual time step because the matrix was at first singular, meaning that the steady-state value of the system, which involves the inverse of **W**, did not exist. By examining the eigenvalues of **W** for each time step it was determined that acceptable performance was obtained when the state response, used to formulate **W**, was first averaged over 50 time steps.

3. Integration of the network following formulation of the **W** and **I** matrices revealed that the outputs did converge to the actual values of the system parameters. Convergence was speeded up to within 2 cycles of the natural frequency by introducing a scale factor of 50 to multiply **W** and **I**.

4. For the network to identify the actual system parameters, the range of the non-linear sigmoid function must encompass the parameters' expected values.

5. Imperfect estimation of parameter matrices when using full-state and reduced-order observers produced unacceptable system-state estimation error which led to incorrect formulations of **W** and **I** and inaccurate network solutions.

6. The network exhibited an acceptable ability to track significant changes in system parameters as time progressed. When the varying gain parameter [**B** (1)] dropped to a value close to zero, however, the system was not being persistently excited and the solution performance degraded drastically. Inaccuracies in tracking over all time intervals pointed out the need

for persistent excitation by the test signal to ensure that **W** is well-conditioned.

C. RECOMMENDATIONS

The following topics are presented for possible future research:

1. The number of time steps in the response of the system over which to average **W** and **I** should be optimized to minimize the network solution time.

2. A routine should be included with the sigmoid function to ensure that its range will always encompass the expected values of the system parameters without being excessively large.

3. Performance of the network should be investigated when the system input signal contains random noise.

4. The value of the gain S1 should be optimized to ensure maximum speed of convergence of the network solution.

5. Additional experimentation with Hopfield networks and observers should be conducted including routines to minimize the error in system state estimation.

6. Practical issues in test signal amplitude and frequency content need to be addressed to ensure the appropriate condition of **W**.

7. Hopfield networks should be implemented for identification of the actual operating parameters of NPS AUV II.

APPENDIX A
PROOF OF STABILITY OF THE HOPFIELD NETWORK FOR PARAMETER IDENTIFICATION

In the system identification problem, the estimation of erroneous parameters leads to an equation error of the kind,

$$\mathbf{e}_j(t) = \mathbf{H}(t)\mathbf{V}_j - \dot{\mathbf{x}}_j(t) \; ; \; j = 1, n$$

in which the system model and the parameter vector so formed are obtained as follows.

System equation: $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \; ; \; \mathbf{y}(t) = \mathbf{C}\,\mathbf{x}(t)$

Parameter vector definition:

$$\mathbf{V}'(t) = [\ldots \mathbf{a}_i \;\; \mathbf{b}_i \ldots] \; i = 1, n \in \Re^{n(n+r)}$$

where a_i and b_i are the i^{th} row of the system parameter matrices, **A** and **B**. If the measurement matrix $\mathbf{H}(t)$, is defined as,

$$\mathbf{H}(t) = \begin{bmatrix} \mathbf{x}'(t)\mathbf{u}'(t) & \ldots 0 \ldots & \ldots . 0 \ldots \\ \ldots 0 \ldots & \mathbf{x}'(t)\mathbf{u}'(t) & \ldots 0 \ldots \\ \ldots 0 \ldots & \ldots 0 \ldots & \mathbf{x}'(t)\mathbf{u}'(t) \end{bmatrix} ; \in \Re^{n \cdot n(n+r)}$$

this leads to

$$\dot{\mathbf{x}}(t) = \mathbf{H}(t)\mathbf{V}(t) + \mathbf{e}(t)$$

in which the errors $\mathbf{e}(t)$ account for measurement noise and errors due to parameter mismatch.

Definition of a positive definite averaged error squared energy function, J, leads to

$$J = A\sum_{j=1}^{n} \{ e_j(t) e_j'(t) \}$$

and in terms of the parameter set $\mathbf{V}(t)$,

$$J = A\{ [\, \mathbf{H}(t)\mathbf{V}(t) - \dot{\mathbf{x}}(t) \,]' [\, \mathbf{H}(t)\mathbf{V}(t) - \dot{\mathbf{x}}(t) \,] \}$$

giving

$$J = \mathbf{V}'(t) A [\mathbf{H}'(t)] \mathbf{H}(t)\mathbf{V}(t) - 2\mathbf{V}'(t) A [\, \mathbf{H}'(t)\dot{\mathbf{x}}'(t) \,] + A [\, \dot{\mathbf{x}}(t)\dot{\mathbf{x}}(t) \,]$$

Since J is a positive definite function of $\mathbf{V}(t)$, convergence of parameters to a stable set in which J is minimum is guaranteed if its time derivative is negative.

The required stability condition is then,

$$\frac{dJ}{dt} = \frac{\partial J}{\partial \mathbf{V}}(t)\dot{\mathbf{V}}(t) < 0 \quad \forall \quad \mathbf{V}(t),\ t = [0,\infty]$$

Such a condition is met if,

$$\dot{\mathbf{V}} = -\mathrm{sgn}\left(\frac{\partial J}{\partial \mathbf{V}} \right)$$

The Hopfield network can be shown to meet that condition since,

$$\frac{\partial J}{\partial \mathbf{V}} = A [\, \mathbf{H}'(t)\mathbf{H}(t) \,]\mathbf{V} - A [\, \mathbf{H}'(t)\dot{\mathbf{x}}(t) \,]$$

or,

$$\frac{\partial J}{\partial \mathbf{V}} = \mathbf{W}\mathbf{V} - \mathbf{I}$$

where, $\mathbf{W} = A [\, \mathbf{H}'(t)\mathbf{H}(t) \,]$, and $\mathbf{I} = A [\, \mathbf{H}'(t)\dot{\mathbf{x}}(t) \,]$

Defining $\frac{\partial J}{\partial \mathbf{V}}(t) = -\frac{d\Theta}{dt}$ leads to the synaptic excitation equation of Hopfield,

$$\frac{d\Theta}{dt} = -\mathbf{W}\mathbf{V} + \mathbf{I} \quad \text{with} \quad \mathbf{V} = g(\Theta)$$

$g(\Theta)$ is always increasing as (Θ) increases (i.e. $g(\Theta)$ lies in the 1st - 3rd quadrant in the θ - $\mathbf{V}$ plane), then $\dot{\mathbf{V}} = g'(\Theta)\frac{d\Theta}{dt}$ with $g'(\Theta)$ always positive. The result demonstrates that J will seek a minimum as

$$\dot{\mathbf{V}}(t) = -g'(\Theta(t))\frac{\partial J}{\partial \mathbf{V}}(t),$$

which meets the stability condition shown.

REALIZATION OF THE SOLUTION

In spite of the above, convergence of $d\Theta/dt$ to 0 as t progresses does not always lead to convergence of $\mathbf{V}(t)$ to the correct parameters, and one additional condition is necessary to impose.

Assume that $\Theta_{ss} \rightarrow [-\infty, \infty]$ and define,

$g_{max} = \lim g(\Theta)$ as $\Theta \rightarrow \infty$ and $g_{min} = \lim g(\Theta)$ as $\Theta \rightarrow -\infty$

Define $\mathbf{V}_{ss}$ as the steady-state solution of $\mathbf{V}(t)$ as t become large.

Define Θ_{ss} as the steady-state solution of $\Theta(t)$ as t becomes large.

It follows that unless

$$g_{min} < \mathbf{V}_{ss} < g_{max}$$

a steady state solution for V does not exist. The point is illustrated in Figure A1.

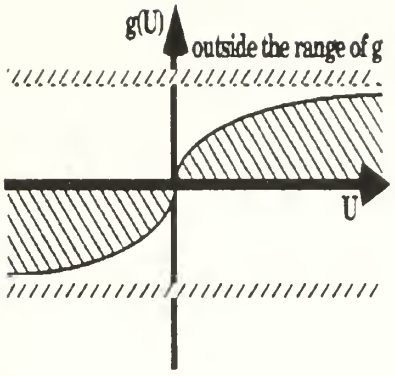

Figure A1. Range of g, U=Θ

APPENDIX B
MAIN PROGRAM FOR HOPFIELD NETWORK WITH LINEAR ACTIVATION FUNCTION

```
%   Main Program for Hopfield network with linear activation
%   function

%   program neu

to=0; tf=1; xo=[1 1];
[tnew,xnew]=ode23('system',to,tf,xo);
n1=50;
t=tnew(1:n1);
x=xnew(1:n1,:);

for i=1:n1
    [ff]=system(t(i),x(i,:)); f(i,:)=ff; end;
u=sin(t); f=f/n1;

xx1=x(:,1)'*x(:,1)/n1;
xx2=x(:,2)'*x(:,2)/n1;
xx12=x(:,1)'*x(:,2)/n1;
x1u=x(:,1)'*u/n1;
x2u=x(:,2)'*u/n1;
uu=u'*u/n1;

%   Calculate average values for weight and bias matrices

W=zeros(6,6); I=zeros(1,6);
W(1,1)=xx1; W(1,2)=xx12; W(1,5)=x1u; W(2,5)=x2u;
W(2,1)=W(1,2);   W(2,2)=xx2;   W(3,3)=xx1;   W(3,4)=W(1,2);
W(3,6)=W(1,5);
W(4,3)=W(2,1);  W(4,4)=W(2,2);  W(4,6)=x2u;  W(5,1)=W(1,5);
W(5,2)=W(2,5);
W(5,5)=uu; W(6,6)=uu; W(6,3)=W(3,6); W(6,4)=W(4,6);
I(1)=x(:,1)'*f(:,1); I(2)=x(:,2)'*f(:,1); I(3)=x(:,1)'*f(:,2);
I(4)=x(:,2)'*f(:,2); I(5)=u'*f(:,1); I(6)=u'*f(:,2);

%   Perform Euler integration on the Hopfield network
%   algorithm.

dt=.02; int=100; s1=50;
tho=ones(1,6);
    for i=2:int;
        [thdot]=hop(W,I,tho,s1);
        th=thdot*dt+tho';
        tho=th'; thpl(i,:)=tho;
```

```
    end;
theta=[-.94248 12.566 -12.566 -.94248 1 2];
thvec=sqrt(theta*theta');
for i=1:100;
    tvec(i)=dt*i;
    thplvec(i)=sqrt(thpl(i,:)*thpl(i,:)');
end;
wo=sqrt(.94248^2+12.566^2);
plot(wo*tvec,thplvec/thvec);


%   Subroutine containing Hopfield algorithm

%   program hop

function[thdot]=hop(W,I,th,s1)

for i=1:6
    V(i)=th(i);
end;

V=[V(1) V(2) V(3) V(4) V(5) V(6)]';
thdot=-s1*W*V+s1*I';



%   Subroutine containing state-space formulation of test case

%   program system

function[f]=system(t,x)

u=sin(t);
A=[-.94248 12.566;-12.566 -.94248]; B=[1;2];
f1=A(1,1)*x(1)+A(1,2)*x(2)+B(1)*u;
f2=A(2,1)*x(1)+A(2,2)*x(2)+B(2)*u;
f=[f1 f2];
```

APPENDIX C
MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION

```
%   Main Program for Hopfield network with sigmoid activation
%   function

%   program neu2sig

to=0; tf=1; xo=[1 1];
[tnew,xnew]=ode23('system',to,tf,xo);
n1=50;
t=tnew(1:n1);
x=xnew(1:n1,:);

for i=1:n1
    [ff]=system(t(i),x(i,:)); f(i,:)=ff; end;
u=sin(t); f=f/n1;

xx1=x(:,1)'*x(:,1)/n1;
xx2=x(:,2)'*x(:,2)/n1;
xx12=x(:,1)'*x(:,2)/n1;
x1u=x(:,1)'*u/n1;
x2u=x(:,2)'*u/n1;
uu=u'*u/n1;

%   Calculate average values for weight and bias matrices

W=zeros(6,6); I=zeros(1,6);
W(1,1)=xx1; W(1,2)=xx12; W(1,5)=x1u; W(2,5)=x2u;
W(2,1)=W(1,2);   W(2,2)=xx2;   W(3,3)=xx1;   W(3,4)=W(1,2);
W(3,6)=W(1,5);
W(4,3)=W(2,1);  W(4,4)=W(2,2);  W(4,6)=x2u;  W(5,1)=W(1,5);
W(5,2)=W(2,5);
W(5,5)=uu; W(6,6)=uu; W(6,3)=W(3,6); W(6,4)=W(4,6);
I(1)=x(:,1)'*f(:,1); I(2)=x(:,2)'*f(:,1); I(3)=x(:,1)'*f(:,2);
I(4)=x(:,2)'*f(:,2); I(5)=u'*f(:,1); I(6)=u'*f(:,2);

%   Perform Euler integration on the Hopfield network
%   algorithm.

dt=.02; int=100; s1=50; lmda=0.1; G=20;
tho=ones(1,6);
    for i=2:int;
        [thdot,V]=hop2sig(W,I,tho,s1,lmda,G);
        th=thdot*dt+tho';
```

```
        tho=th'; thpl(i,:)=tho; tvec(i)=dt*i;
        Vpl(i,:)=V'; thss(i,:)=(inv(-W)*I')';
    end;
theta=[-.94248 12.566 -12.566 -.94248 1 2];
thvec=sqrt(theta*theta');
for i=1:6;
    VVpl(:,i)=Vpl(:,i)/theta(i);
end;
wo=sqrt(.94248^2+12.566^2);
plot(wo*tvec,VVpl);


%   Subroutine containing Hopfield algorithm

%   program hop2sig

function[thdot,V]=hop2sig(W,I,th,s1,lmda,G)

for i=1:6
    V(i)=G*(2/(1+exp(-lmda*th(i)))-1);
end;

V=[V(1) V(2) V(3) V(4) V(5) V(6)]';
thdot=-s1*W*V+s1*I';



%   Subroutine containing state-space formulation of test case

%   program system

function[f]=system(t,x)

u=sin(t);
A=[-.94248 12.566;-12.566 -.94248]; B=[1;2];
f1=A(1,1)*x(1)+A(1,2)*x(2)+B(1)*u;
f2=A(2,1)*x(1)+A(2,2)*x(2)+B(2)*u;
f=[f1 f2];
```

APPENDIX D
MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION AND FULL-STATE OBSERVER

```
%   Main Program for Hopfield network with sigmoid activation
%   function and full-state observer

%   program neuobs
%   Development of state variable time history

to=0; tf=1; xo=[1 1 1 1];
[tnew,xnew]=ode23('observer',to,tf,xo);
n1=50;
t=tnew(1:n1);
x=xnew(1:n1,:);

for i=1:n1
    [ff]=observer(t(i),x(i,:)); f(i,:)=ff; end;
u=sin(t); f=f(:,3:4)/n1;

xx1=x(:,3)'*x(:,3)/n1;
xx2=x(:,4)'*x(:,4)/n1;
xx12=x(:,3)'*x(:,4)/n1;
x1u=x(:,3)'*u/n1;
x2u=x(:,4)'*u/n1;
uu=u'*u/n1;

%   Calculate average values for weight and bias matrices

W=zeros(6,6); I=zeros(1,6);
W(1,1)=xx1; W(1,2)=xx12; W(1,5)=x1u; W(2,5)=x2u;
W(2,1)=W(1,2);   W(2,2)=xx2;   W(3,3)=xx1;   W(3,4)=W(1,2);
W(3,6)=W(1,5);
W(4,3)=W(2,1);  W(4,4)=W(2,2);  W(4,6)=x2u;  W(5,1)=W(1,5);
W(5,2)=W(2,5);
W(5,5)=uu; W(6,6)=uu; W(6,3)=W(3,6); W(6,4)=W(4,6);
I(1)=x(:,1)'*f(:,1); I(2)=x(:,2)'*f(:,1); I(3)=x(:,1)'*f(:,2);
I(4)=x(:,2)'*f(:,2); I(5)=u'*f(:,1); I(6)=u'*f(:,2);

%   Perform Euler integration on the Hopfield network
%   algorithm.

dt=.02; int=100; s1=50; lmda=0.1; G=20;
```

```
tho=zeros(1,6);
    for i=2:int;
        [thdot,V]=hopobs(W,I,tho,s1,lmda,G);
        th=thdot*dt+tho';
        tho=th'; thpl(i,:)=tho; tvec(i)=dt*i;
        Vpl(i,:)=V'; thss(i,:)=(inv(-W)*I')';
    end;
theta=[-.94248 12.566 -12.566 -.94248 1 2];
thvec=sqrt(theta*theta');
for i=1:6;
    VVpl(:,i)=Vpl(:,i)/theta(i);
end;
wo=sqrt(.94248^2+12.566^2); plot(wo*tvec,VVpl);


%   Subroutine containing Hopfield algorithm

%   program hopobs

function[thdot,V]=hopobs(W,I,th,s1,lmda,G)

for i=1:6
    V(i)=G*(2/(1+exp(-lmda*th(i)))-1);
end;

V=[V(1) V(2) V(3) V(4) V(5) V(6)]';
thdot=-s1*W*V+s1*I';



%   Subroutine containing state-space formulation of test case
%   and full-state observer

%   program observer

function[f]=observer(t,x)

u=sin(t);
A=[-.94248 12.566;-12.566 -.94248]; B=[1;2]; C=[0 1];
Ao=[-.7 10;-10 -.7]; Bo=[2.44;1.39]; Co=[0 1];
%   Ao=A; Bo=B;
Q=eye(2); R=.01;

xx=x(1:2)'; xhat=x(3:4)';

[K,s]=lqr(Ao,Bo,Q,R);

xxdot=A*xx+B*u;
xhatdot=(Ao-K'*Co)*xhat+Bo*u+K'*Co*xx;

f=[xxdot' xhatdot'];
```

APPENDIX E
MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION AND REDUCED-ORDER OBSERVER

```
%   Main Program for Hopfield network with sigmoid activation
%   function and reduced-order observer

%   program neuobs
%   Development of state variable time history

to=0; tf=1; xo=[1 1 1];
[tnew,xnew]=ode23('observer',to,tf,xo);
n1=50;
t=tnew(1:n1);
x=xnew(1:n1,:);
for i=1:n1
    [ff]=observer(t(i),x(i,:));
    f(i,1)=ff(3); f(i,2)=ff(2);
end;
u=sin(t); f=f/n1;

xx1=x(:,3)'*x(:,3)/n1;
xx2=x(:,2)'*x(:,2)/n1;
xx12=x(:,3)'*x(:,2)/n1;
x1u=x(:,3)'*u/n1;
x2u=x(:,2)'*u/n1;
uu=u'*u/n1;

%   Calculate average values for weight and bias matrices

W=zeros(6,6); I=zeros(1,6);
W(1,1)=xx1; W(1,2)=xx12; W(1,5)=x1u; W(2,5)=x2u;
W(2,1)=W(1,2);   W(2,2)=xx2;   W(3,3)=xx1;   W(3,4)=W(1,2);
W(3,6)=W(1,5);
W(4,3)=W(2,1);  W(4,4)=W(2,2);  W(4,6)=x2u;  W(5,1)=W(1,5);
W(5,2)=W(2,5);
W(5,5)=uu; W(6,6)=uu; W(6,3)=W(3,6); W(6,4)=W(4,6);
I(1)=x(:,1)'*f(:,1); I(2)=x(:,2)'*f(:,1); I(3)=x(:,1)'*f(:,2);
I(4)=x(:,2)'*f(:,2); I(5)=u'*f(:,1); I(6)=u'*f(:,2);

%   Perform Euler integration on the Hopfield network
%   algorithm.

dt=.02; int=100; s1=50; lmda=0.1; G=20;
tho=zeros(1,6);
```

```
    for i=2:int;
        [thdot,V]=hopobs(W,I,tho,s1,lmda,G);
        th=thdot*dt+tho';
        tho=th'; thpl(i,:)=tho; tvec(i)=dt*i;
        Vpl(i,:)=V'; thss(i,:)=(inv(-W)*I')';
    end;
theta=[-.94248 12.566 -12.566 -.94248 1 2];
thvec=sqrt(theta*theta');
for i=1:6;
    VVpl(:,i)=Vpl(:,i)/theta(i); end;
wo=sqrt(.94248^2+12.566^2); plot(wo*tvec,VVpl);


%   Subroutine containing Hopfield algorithm

%   program hopobs

function[thdot,V]=hopobs(W,I,th,s1,lmda,G)

for i=1:6
    V(i)=G*(2/(1+exp(-lmda*th(i)))-1);
end;

V=[V(1) V(2) V(3) V(4) V(5) V(6)]';
thdot=-s1*W*V+s1*I';



%   Subroutine containing state-space formulation of test case
%   and reduced-order observer

%   program observer

function[f]=observer(t,x)

u=sin(t);
A=[-.94248 12.566;-12.566 -.94248]; B=[1;2]; C=[0 1];
Ao=[-.7 10;-10 -.7]; Bo=[2.44;1.39]; Co=[0 1];
%   Ao=A; Bo=B;
Q=eye(1); R=.01;

xx=x(1:2)'; x1hat=x(3);

xxdot=A*xx+B*u;
x1hatdot=Ao(1,2)*x(2)+Ao(1,1)*x1hat+Bo(1)*u;

f=[xxdot' x1hatdot];
```

APPENDIX F
MAIN PROGRAM FOR HOPFIELD NETWORK WITH SIGMOID ACTIVATION FUNCTION TRACKING VARIATION IN GAIN PARAMETER B(1)

```
%   Main Program for Hopfield network with sigmoid activation
%   function tracking variation in gain parameter B(1)

%   program neuobs1

to=0; tf=1; xo=[1 1];
for a=1:5;
        if a==1; to=0; tf=1;
    elseif a==2; to=1; tf=2;
    elseif a==3; to=2; tf=3;
    elseif a==4; to=3; tf=4;
    elseif a==5; to=4; tf=5;
end;

[t,x]=ode23('system',to,tf,xo);
[n1,p]=size(t); xo=x(n1,:);

for i=1:n1
    [ff]=system(t(i),x(i,:)); f(i,:)=ff; end;
u=sin(t); f=f/n1;

xx1=x(:,1)'*x(:,1)/n1;
xx2=x(:,2)'*x(:,2)/n1;
xx12=x(:,1)'*x(:,2)/n1;
x1u=x(:,1)'*u/n1;
x2u=x(:,2)'*u/n1;
uu=u'*u/n1;

%   Calculate average values for weight and bias matrices

W=zeros(6,6); I=zeros(1,6);
W(1,1)=xx1; W(1,2)=xx12; W(1,5)=x1u; W(2,5)=x2u;
W(2,1)=W(1,2);   W(2,2)=xx2;   W(3,3)=xx1;   W(3,4)=W(1,2);
W(3,6)=W(1,5);
W(4,3)=W(2,1);  W(4,4)=W(2,2);  W(4,6)=x2u;  W(5,1)=W(1,5);
W(5,2)=W(2,5);
W(5,5)=uu; W(6,6)=uu; W(6,3)=W(3,6); W(6,4)=W(4,6);
I(1)=x(:,1)'*f(:,1); I(2)=x(:,2)'*f(:,1); I(3)=x(:,1)'*f(:,2);
I(4)=x(:,2)'*f(:,2); I(5)=u'*f(:,1); I(6)=u'*f(:,2);

%   Perform Euler integration on the Hopfield network
```

```
%   algorithm.

dt=.02; int=1000; s1=50; lmda=0.1; G=20;
tho=ones(1,6);
    for i=2:int;
        [thdot,V]=hopobs(W,I,tho,s1,lmda,G);
        th=thdot*dt+tho';
        tho=th'; thpl(i,:)=tho; Vpl(i,:)=V'; end;
theta=[-.94248 12.566 -12.566 -.94248 1 2];
thvec=sqrt(theta*theta');
wo=sqrt(.94248^2+12.566^2);
tvec=dt*[1:5000]';

gainb(:,a)=Vpl(:,5);
for ig=1:5;
    j=1000*ig;
    k=1000*ig-999;
    gb(k:j)=gainb(:,ig);
end;

for    i=1:1000; BB(i)=1.0;  end;
for i=1001:2000; BB(i)=5.0;  end;
for i=2001:3000; BB(i)=10.0; end;
for i=3001:4000; BB(i)=0.5;  end;
for i=4001:5000; BB(i)=-5.0; end;

plot(wo*tvec,BB,wo*tvec,gb);


%   Subroutine containing Hopfield algorithm

%   program hopobs

function[thdot,V]=hop(W,I,th,s1,lmda,G)

for i=1:6
    V(i)=G*(2/(1+exp(-lmda*th(i)))-1);
end;

V=[V(1) V(2) V(3) V(4) V(5) V(6)]';
thdot=-s1*W*V+s1*I';



%   Subroutine containing state-space formulation of test case
%   with step changes in gain parameter B(1)

%   program system

function[f]=system(t,x)
```

```
u=sin(t);
A=[-.94248 12.566;-12.566 -.94248]; B=[1;2];

if     t>=1.0 & t<2.0; B=[5;2];
elseif t>=2.0 & t<3.0; B=[10;2];
elseif t>=3.0 & t<4.0; B=[0.5;2];
elseif t>=4.0 & t<5.0; B=[-5.0;2];
end;

xx=x(1:2)';
xxdot=A*xx+B*u;
f=xxdot';
```

LIST OF REFERENCES

1. Healey, A.J., and others, "Mission Planning, Execution, and Data Analysis for the NPS AUV II Autonomous Underwater Vehicle," *Proceedings* of 1st NSF Workshop on Mobile Robots for Subsea Environments, Monterey, California, 23 October 1990, pp. 177-186.

2. McCulloch, W.S., and Pitts, W.H., "A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics,* v. 5, pp. 115-133, 1943.

3. Dietz, W.E., Kiech, E.L., and Ali, M., "Jet and Rocket Engine Fault Diagnostics in Real Time," *Journal of Neural Network Computing,* v. 1, no. 1, 1989.

4. Shoureshi, R., and Chu, R., "Neural Space Representation of Dynamical Systems," *Intelligent Control Systems,* ASME Bound Volume DSC-v. 16, presented at 1989 Winter Annual Meeting, pp. 63-68, 1989.

5. Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences, U.S.A.*, v. 79, pp. 2554-2558, April 1982.

6. Hopfield, J.J., "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proceedings of the National Academy of Sciences, U.S.A.*, v. 81, pp. 3088-3092, May 1984.

7. Wasserman, P.D., *Neural Computing,* Van Nostrand Reinhold, 1989.

8. Neural Ware, Inc., *Neural Works Professional II: Neural Computing,* 1989.

9. Rosenblatt, R., *Principles of Neurodynamics,* Spartan Books, 1959.

10. Widrow, B., "Adaptive Sampled-Data Systems, a Statistical Theory of Adaptation," *IRE WESCON Convention Record,* part 4, Institute of Radio Engineers, 1959.

11. Minsky, M.L., and Papert, S., *Perceptrons,* MIT Press, 1969.

12. Hebb, D.O., *The Organization of Behavior,* Wiley, 1949.

13. Cohen, M.A., and Grossberg, S.G., "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks, *IEEE Transactions on Systems, Man, and Cybernetics,* v. 13, pp. 815-826, 1983.

14. Friedland, B., *Control System Design,* McGraw-Hill Book Company, pp. 259-289, 1986.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code ME/Hy Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5000	2
4.	Dr. G. Dobeck, Code 4210 Naval Coastal Systems Command Panama City, FL 32407-5000	1
5.	Robert Wilson Head, Systems Engineering Branch DTRC, Carderock Bethesda, MD 20084-5000	1
6.	RADM Evans, Code SEA 92 Naval Sea Systems Command Washington, D.C. 20362	1
7.	Dan Steiger Marine Systems Group Naval Research Laboratory Washington, D.C. 20032	1
8.	Jennifer Rau, Code U25 Naval Surface Weapons Center Silver Spring, MD 20903-5000	1
9.	Naval Engineering Curricular Office, Code 34 Naval Postgraduate School Monterey, CA 93943-5000	1
10.	Mr. Glenn Reid, Code U401 Naval Surface Warfare Center Silver Spring, MD 20901	1

11. Commandant (G-MTH-2) 2
U.S. Coast Guard
2100 Second Street S.W.
Washington, D.C. 20593-0001

28 JUN 93

59348

NOV - 1 1994

4

PRINTED IN U.S.A.

Keep this card in the book pocket
Book is due on the latest date stamped

Thesis
M3564
c.1

Marsilio

Use of Hopfield networks for system identification and failure detection in autonomous underwater vehicles.

DUDLEY KNOX LIBRARY
3 2768 00018385 9